



SYSTEMS ENGINEERING
Research Center

Prototype of a Graphical CONOPS (Concept of Operations) Development Environment for Agile Systems Engineering

Final Technical Report SERC-2012-TR-030

March 23, 2012

Principal Investigator - Dr. Robert Cloutier - Stevens Institute of Technology
Co-Principal Investigator: Dr. Sara McComb - Purdue University

Team Members

Dr. Abhijit Deshmukh, Senior Researcher - Purdue University
Dr. Teresa Zigh, Senior Researcher - Stevens Institute of Technology
Peter Korfiatis, Research Assistant - Stevens Institute of Technology
Behnam Esfahbod, Research Assistant - Stevens Institute of Technology
Peizhu Zhang, Research Assistant - Stevens Institute of Technology
Keith Hall, Research Assistant - Purdue University

Report Documentation Page		Form Approved OMB No. 0704-0188
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.		
1. REPORT DATE 23 MAR 2012	2. REPORT TYPE Final	3. DATES COVERED
4. TITLE AND SUBTITLE Prototype of a Graphical CONOPS (Concept of Operations) Development Environment for Agile Systems Engineering		5a. CONTRACT NUMBER H98230-08-D-0171
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Cloutier /Dr. Robert		5d. PROJECT NUMBER RT 30
		5e. TASK NUMBER DO1 TTO2
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Stevens Institute of Technology Purdue University		8. PERFORMING ORGANIZATION REPORT NUMBER SERC-2012-TR-030
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) DASD (SE)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release, distribution unlimited.		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT Gaming and immersive virtual environments provide a new way to engage stakeholders during early stages of Systems Engineering lifecycle to help them reach a common mental model of the concept of operations. A weak link in the Systems Engineering lifecycle is often the connection between what the users need and what the system developers think the users need, together with a shared understanding of the operational environment and associated constraints and dependencies. The current system development environment calls for user needs to be specified in a Concept of Operations (CONOPS) document, which provides a foundation of future system capabilities and describes typical scenarios that it will encounter. Given the size and complexity of today's systems, CONOPS development can take considerable time and effort, which can cause its production to be incomplete and insufficient. This introduces misunderstanding and miscommunication early in the Systems Engineering lifecycle. This paper describes a method allowing stakeholders to express their needs through a model-based approach to create a graphical CONOPS leveraging gaming technology. The resulting CONOPS would provide system developers with direct access to the needs of stakeholders, and would enable the creation of Model-Based Systems Engineering artifacts early in the development lifecycle. This work is exploring the use of an Integrated Concept Engineering System to improve the CONOPS development process.		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 92	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

Copyright © 2012 Stevens Institute of Technology, Systems Engineering Research Center

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

NO WARRANTY

THIS STEVENS INSTITUTE OF TECHNOLOGY AND SYSTEMS ENGINEERING RESEARCH CENTER MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. STEVENS INSTITUTE OF TECHNOLOGY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. STEVENS INSTITUTE OF TECHNOLOGY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution except as restricted below.

Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use:* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Systems Engineering Research Center at dschultz@stevens.edu

* These restrictions do not apply to U.S. government entities.

This page intentionally left blank

ABSTRACT

Gaming and immersive virtual environments provide a new way to engage stakeholders during early stages of Systems Engineering lifecycle to help them reach a common mental model of the concept of operations. A weak link in the Systems Engineering lifecycle is often the connection between what the users need and what the system developers think the users need, together with a shared understanding of the operational environment and associated constraints and dependencies. The current system development environment calls for user needs to be specified in a Concept of Operations (CONOPS) document, which provides a foundation of future system capabilities and describes typical scenarios that it will encounter. Given the size and complexity of today's systems, CONOPS development can take considerable time and effort, which can cause its production to be incomplete and insufficient. This introduces misunderstanding and miscommunication early in the Systems Engineering lifecycle. This paper describes a method allowing stakeholders to express their needs through a model-based approach to create a graphical CONOPS leveraging gaming technology. The resulting CONOPS would provide system developers with direct access to the needs of stakeholders, and would enable the creation of Model-Based Systems Engineering artifacts early in the development lifecycle. This work is exploring the use of an Integrated Concept Engineering System to improve the CONOPS development process.

This page intentionally left blank

TABLE OF CONTENTS

Abstract	3
Table of Contents	5
Figures and Tables	7
1 Summary	9
2 Introduction.....	10
2.1 Concept Engineering System (CES) Vision	11
2.2 Development Environment	12
2.3 Development Process.....	16
3 Work Performed	17
3.1 CES Prototype	17
3.1.1 Modeling the CES Prototype.....	19
3.2 Prototype First Release	26
3.3 Usability Planning.....	29
4 Workshop	30
4.1 Scenarios	30
4.1.1 News Agency Scenario 1.....	31
4.1.2 News Agency Scenario 2.....	31
4.1.3 News Agency Scenario 3.....	31
4.1.4 News Agency Scenario 4.....	32
4.1.5 Primitive Library	32
4.2 Workshop Results.....	38
4.2.1 Evaluator feedback.....	38
5 Lessons Learned	40
5.1 Research Lessons/Questions	40
5.2 Project Management	40
5.3 System/Software Architecture	42
5.4 Project/Code Construction.....	43
6 Conclusions.....	45
7 Recommendations for Continuation.....	46
Appendices	49
Appendix A - Usability	49
A.1 Initial.....	49
A.2 Mid-Development.....	52
A.3 End Development	63
A.4 Discussion and recommendatons	67
A.5 Heuristic Evaluation - A System Checklist.....	70
A.5 REFERENCES	90

Appendix B: References..... 91

FIGURES AND TABLES

Figure 1: Evaluation of Serious Gaming Technologies	14
Figure 2: Rational Unified Process for software development.....	16
Figure 3: CES Conceptual Architecture	18
Figure 4: CES Prototype Top Level Use Case Diagram	19
Figure 5: CES Prototype Manage Primitives Use Case.....	19
Figure 6: CES Prototype Manage Scenarios Use Case.....	20
Figure 7: CES Prototype Manage CONOPS Use Case.....	20
Figure 8: CES Prototype Develop Scenarios Use Case	20
Figure 9: CES Prototype Produce Documentation Use Case	20
Figure 10: CES Prototype Manage Primitives Activities	22
Figure 11: CES Prototype Develop Scenario Activities	23
Figure 12: CES Prototype Assemble Scenario Activities	23
Figure 13: CES Prototype Logical Architecture	24
Figure 14: CES Prototype Logical Internal Architecture	25
Figure 15: CES Prototype Start Screen	26
Figure 16: CES Prototype Developer Interface	26
Figure 17: CES Prototype Author Interface	27
Figure 18: CES Prototype: Adding Primitives	27
Figure 19: CES Prototype: Connecting Primitives.....	28
Figure 20: CES Prototype: Changing Scenes.....	28
Figure 21: News Agency Scenario 1.....	33
Figure 22: News Agency Scenario 2.....	34
Figure 23: News Agency Scenario 3.....	35
Figure 24: News Agency Scenario 4.....	36
Figure 25: News Agency Scenario Primitives	37

Table 1: Game Development Engines	12
Table 2: RT30 Sponsor Meeting Schedule.....	17
Table 3: Workshop Participant Data	30
Table 4: CES Evaluator Feedback	38
Table 5: CES Prototype Minor Improvements	46
Table 6: CES Prototype Major Improvements.....	47

1 SUMMARY

The work performed to date for this research task was designed to understand the state of the practice for Concept of Operations (CONOPS) development. It found that there are no significant barriers to using 3D gaming technologies to producing a tool which may assist teams performing concept engineering.

This research task was designed to begin investigating the five orthogonal dimensions of cognition: things, places, paths, actions, and causes suggested as key by cognitive linguists. The goal of this research task was to produce a limited functionality concept engineering prototype to enable the creation of graphical CONOPS.

To understand the software's functionality and its applicability to address a CONOPS problem, a workshop was conducted with personnel selected by the sponsor. Feedback from this workshop is incorporated into this report.

The generated prototype supports an analyst with a basic set of graphical primitives which are selected and arranged in a manner which will tell a story by way of scenes collected into scenarios. During this research project, it became clear that a solid application infrastructure to provide code stability if any reliable research would be performed in the future. Therefore, much of the work for this effort was expended in designing and building that infrastructure. The research team feels that application infrastructure is sound, and positions the research for the next phase.

2 INTRODUCTION

This RT0030 is a follow-on task to previous research (RT0003) - Investigation of a Graphical CONOPS (Concept of Operations) Development Environment for Agile Systems Engineering. RT0003 can be summarized as research into the state of the practice for current approaches to Concept of Operations (CONOPS) development in use in various DoD and commercial organizations. RT 0030 was initiated to explore the use of computerized environments to improve the process of CONOPS development.

The scope of this research task was summarized in the kick-off meeting held on January 24, 2011 as:

- The goal of RT30 is to produce a limited functionality concept engineering system prototype to enable the creation of graphical CONOPS.
- The prototype will leverage the proposed agile process defined in RT3 phase 1 of the research, and organize the primitives derived in the research performed in RT3 phase 2.
- The prototype will provide an analyst a set of graphical primitives which can be selected and arranged in a manner which will tell a story. When the analyst is satisfied the story is correct, the prototype will convert this story into a textual document.
- The analyst will also be able to add textual “boilerplate” such as organization, goals, etc. that will be incorporated in an auto-generated CONOPS document.

In addition to RT 030, RT031 also evolved from the initial research task, RT003. Since RT031 seeks to address related sponsor defined needs, the research team defined a high level research question to tie together the RT 003/030/031 thread:

Can the process of Concept Engineering improve the understanding and development of a concept of operations using gaming technologies along with an interactive, collaborative, and graphical environment?

From this question, each research task contains lower level research questions to address specific task goals. For RT 030, these include:

- Can the process of CONOPS development and understanding be improved through the use of a “drag and drop” graphical user interface?
- Can real-time collaboration between distributed stakeholders improve the CONOPS development?
- Can a real-time collaboration environment enable quicker consensus on CONOPS generation?

- Does a shared mental model improve the communication among stakeholders? Do visual models allow domain-specific stakeholders to better communicate the needed operational needs?
- Will an immersive environment support non-real-time, but rather just-in-time asynchronous collaboration?
- Does 4D (3D + time) provide deeper insights into the operational concepts of a proposed system than traditional textual documents or static 2D story boarding?

It is important to note that this research task is funded as basic research, and not a software development project. In this case, software is being developed to address the research questions defined above. As such, capabilities and functionality of any software developed will be closely tied to these and any future research questions.

2.1 CONCEPT ENGINEERING SYSTEM (CES) VISION

The approach taken to meet the goals of this research is twofold. First, as developed in the RT 003 report (Cloutier et al., 2009) and published in (Mostashari, McComb, Kennedy, Cloutier, & Korfiatis, 2011) a process (Agile CONOPS Development process) has been created to guide systems developers and users through development of a CONOPS. The Concept Engineering System (CES) is a software tool intended to provide developers with a virtual environment in which a graphical CONOPS can be collaboratively created.

The vision of CES and its contribution to CONOPS development includes the following top level objectives:

- Ability to engage non-technical stakeholders in the graphical CONOPS development process through use of a drag and drop interface
- Enhancement of user and developer collaboration during graphical CONOPS development, both synchronously and asynchronously (text, speech, video and/or drawings)
- Ability to incorporate logic-based simulation (agent-based, discrete event, system dynamics, and physical systems) and physics-based simulation (AutoCad®, SolidWorks®) capabilities
- Ability to develop user-directed visualizations dynamically created based on graphical CONOPS data

Based on discussions with the sponsor and identification of current CONOPS process shortcomings, some of the features and expectations of CES include:

- Ease of use for non-technical users

- Ability to create new objects by developers, experts (modelers) or authors
- Intuitive environment enabling authors to quickly compose a system using a drag and drop interface
- Multi-modal interface (e.g. touch screen enabled)
- Distributed, multi-user interaction
- Simulation basis for all objects with emphasis on inherent object interaction and ability for users to perform simulations in real time and view concise reports
- Graphical programming, 3D creation and viewing
- Output includes textual CONOPS

Given the goals of this project, and the RT003 examination of technologies applicable towards this research, a game development environment was chosen to enable user interaction for CES. Gaming technologies have been used extensively to solve complex problems, and have been well applied in the defense and intelligence domains for training. However, their use has not been well explored in system development, and specifically during early conceptual design. As such, much thought was placed in choosing the proper game development engine for the task at hand. The evaluation process and choice of game development engine is recounted below.

2.2 DEVELOPMENT ENVIRONMENT

In early 2011, under RT 003, gaming technology was investigated as the core backbone link between all the CONOPS-specifics functionality – including scenario-building, simulation using various third-party vendor packages, and generating SysML/XML output from vendor offerings already in use by soldiers in the field. To determine which platform to select, a broad range of available gaming environments were examined, listed in Table 1.

Table 1: Game Development Engines

Torque 2D	Unreal DK	Vicious
Torque 3D	ID Tech (Doom 3)	Open Simulator
Quest 3D	Cry Engineer	C4
Unity	MS-XNA	Gamebryo
Unity Pro	Adobe Flash	Dark Basic
Unreal Engine	Source	Open Simulator

The survey examined qualitative evaluation of each platform on a number of criteria within several overall categories, as shown below:

Features/Capabilities

- Multiplayer
- 3D/2D representations
- Specific comparative strengths and limitations

- Development languages and physics engines supported

Deployment

- Client-Server capability
- Web, PC, Mac supportable
- Minimum CPU and RAM required
- Video card
- Minimum bandwidth

Compatibility with Open Source

- Source code
- Open source components
- Open interfaces

Cost:

- per seat
- to deploy
- license specifications

The evaluations of the software packages/environments along these dimensions are shown in the figure below.

NAME	FEATURES/CAPABILITIES					DEPLOYMENT									OPEN SOURCE FRIENDLY			COST		
Tech	multiplayer	3D/2D	Strengths	Limitations	development features	Client-Server	Web	PC	Mac	OS	min. CPU	min. RAM	video card	min net BW	source code	open source components	Open interfaces	cost per seat	Cost for deployment	already own
Torque 3D																		\$1000-more for externally funded?		
	yes	3D		support, no dev exp	Physics, art pipeline		yes	yes	yes						yes (extra)	Supports all formats, Blender, Maya				yes
Torque 2D																		\$1,250		yes (source)
	yes	2D			Physics, C++ like scripting		yes	yes	yes						yes (extra)					
Quest 3D							yes	yes	no						?			\$3,150		yes
	yes	3D																		
Unity					Plugin, standalone, physics engine, texture, shading, javascript, C#, Boo (Python)													Free To < \$100k		
	yes	3D & 2D					yes	no	no						no	AV Codec (ogg), .Net, Server add-on				yes
Unity Pro					Plugin, standalone, physics engine, texture, shading, javascript, C#, Boo (Python)													\$1,200		
	yes	3D & 2D	experience, community				yes	yes	yes						yes (extra)	AV Codec (ogg), .Net, Server add-on				yes
Unreal Engine					Physics engine, lighting, shadows, C++													> \$700k		
	yes	3D & 2D		1st person shooter, cost			no	yes	no						yes					no
Unreal DK					Physics engine, lighting, shadows, C++													\$2500/seat/year		no
	yes	3D & 2D		1st person shooter			no	yes	no						no					
CryEngine																		Free (educational / research)		
	?	3D		cost, hardware requirements			yes	yes	no						yes					no
MS XNA					.Net, DirectX, asset pipeline, rendering													free, membership required to play		
	yes	3D & 2D	free	public, users must pay			no	yes	no						no					no
Adobe Flash																				
	yes	2D	ubiquitous plugin	2D, not physics based			yes	yes	yes						no, but projects can be			\$449		yes
Source					Direct3D / OpenGL rendering, facial animation, skeletal animation, physics engine													?		
	yes	3D		cost, 1st person shooter			no	yes	yes						yes					no
Vicious					libraries, strong support for AI													?		no
	yes	3D & 2D					no	yes	no						yes					
IdTech (Doom3)					bump and normal mapping, skeletal animation													?		no
	yes	3D		cost, 1st person shooter			no	yes	yes						no (possibly in 2011)					no
Gamebryo					rapid prototyping, extensible infrastructure, scripting tools													?		
	yes	3D	RPG	likely cost			no	yes	no						yes (extra)					no
Dark Basic					simple scripting of DirectX, camera, lighting, library of commands															
	yes	3D & 2D		limited community, assets			no	yes	no						no			\$40		yes
Open Simulator					physics simulation, multiple engines, multiple clients and protocols															
	yes	3D	open source	not a full engine, alpi			yes	yes	yes						yes			free		no

Figure 1: Evaluation of Serious Gaming Technologies

Of all the criteria evaluated, several dominated the decision-making process; most of these concerned development and deployment. These included (in no particular order):

- an active and responsive user community,
- the ability to port to different platforms easily,
- the ability to easily support multiple developers,
- providing code control (though this is not a production environment),
- supporting a diversity of programming languages transparently, and
- the ability to either have or incorporate open source components.

In today's environment of flat defense budgets, cost is also a factor, although site-wide and server licenses may help mitigate concerns that per seat licenses may incur.

Although not stated as one of the “critical” components of the decision-making process, the availability of scalable 3D models was also crucial. The applications will be operating in (and as) a visually-based immersive environment; having the models and simulation as realistic as possible will help increase the probability of acceptance and usage by the eventual field users. 3D models can also have a considerable cost factor. For this task, the group utilized 3D models that were found at no cost, although the eventual selected platform does have extensive libraries of 3D models, some available at no cost or for a nominal fee.

As can be seen in Figure 1, many of the investigated platforms have major drawbacks (shown in red). Chief among these was their inability to deploy on the Web. A secondary consideration for this phase of our research task is the ability of the tool to interface with open source code and components.

The selected platform was Unity 3D Pro. The learning curve for developers was found to be less daunting than that of most of the other platforms, being more intuitive and the facility to develop and deploy components was relatively easily-acquired.

Unity 3D Pro has an asset server which acts as a central code storage and a rudimentary code control mechanism. It has a rich library of models, environments, scripts, and other development components available, either free or at a nominal cost.

Unity 3D Pro supports a number of programming languages: C#, Boo (Python), and Javascript. The Unity physics engine supports movement, collision, gravity for solid objects, and users can modify textures/meshes. This ability will be critical if terrain generation from various USGS databases is to be evaluated.

Unity 3D Pro has a large user community which is extremely responsive to posted questions, and a forum containing posted solutions to many commonly-found problems or desired effects. As this research task was focused mainly on interfaces between 3rd-

party software, we did not find solutions in user community resources for these tasks, however the resources did help when implementing some of the more complex model representations and movement.

2.3 DEVELOPMENT PROCESS

At the onset of RT0030, it was suggested that the Rational Unified Process (RUP) be used as a framework for the CES software development effort. Figure 2 shows the basic phases of RUP, and the effort required in each phase by development activities.

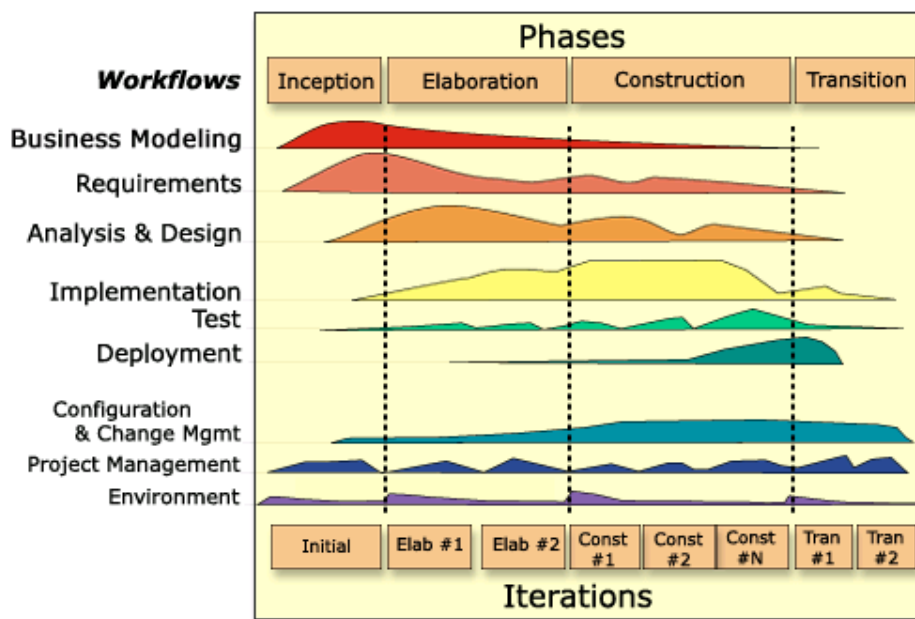


Figure 2: Rational Unified Process for software development

Following RUP has been shown to lead to a large number of software development successes, however in the case of this project it proved to be ineffective. The primary reason for its discontinued use was due to the academic environment the research team was working within. As will be detailed below, there were a number of lessons learned when developing a software product using a primarily student based workforce.

To replace RUP, the research team has look towards a Spiral development type effort. Software releases are made every other week, integrating the most stable form of improvements made to CES. During the two week development cycle, new functionality is implemented while the previous release capabilities are tested by both programmers and non-programmers on the research team. During the next phase of this research, two week releases will remain internal to the research team, but it is hoped that more substantial releases will be made and distributed to the sponsors and their representatives at regular intervals.

3 WORK PERFORMED

The work performed during this research task includes:

- Continued assessment of the current CONOPS development
- CES architectural development and modeling
- CES prototype software development
- Workshop created and executed using sponsor selected personnel to test the CES software, validate the CONOPS improvement line of research and gather evaluator feedback.
- Redefinition of research questions applicable to the research task
- Recommendations for future research and software development efforts for continuation of this research task.

The research team kept sponsors updates to task progress through a number of predetermined meetings, as displayed in Table 2. Additionally, a working meeting was held every week between the members of the research team and a representative from the sponsor to discuss research and development progress, issues and goals.

Table 2: RT30 Sponsor Meeting Schedule

Project Kickoff	1/24/2011
IPR 1	5/12/2011
IPR2	8/5/2011
Pre Workshop Review	11/30/2011
Workshop	12/8/2011

3.1 CES PROTOTYPE

The goal of CES is the development of a tool to enable the investigation of RT 0030 research questions. The prototype of CES is meant to act as a proof of concept demonstration of the applicability of leveraging gaming technologies and virtual environments towards solving complex issues faced during concept engineering.

A high level conceptual architecture for CES can be seen in Figure 3.

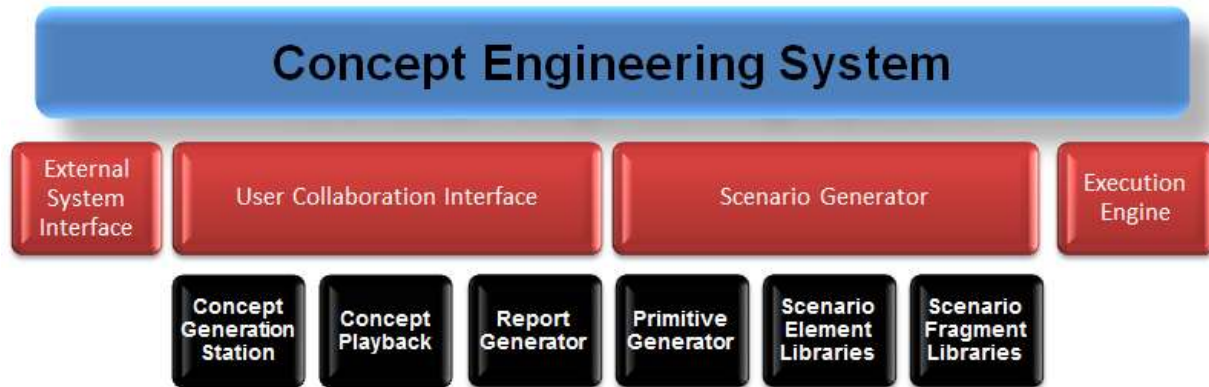


Figure 3: CES Conceptual Architecture

The main components of CES include:

- Concept Generator - user interface enabling the creation of the graphical CONOPS
- Primitive, Scenario and Scenario Fragment Databases - storing primitives; promoting their reuse; storing graphically created scenarios as models
- Primitive Creator – new primitive creation; attribute and 3D visualizations association
- Report Generator – automatic CONOPS document creation to fulfill current contracting requirements
- Concept Playback – allowing development and display of animations reflecting the scenario
- Describe the time components, scenes developed and storyboarded as in movie industry.

3.1.1 MODELING THE CES PROTOTYPE

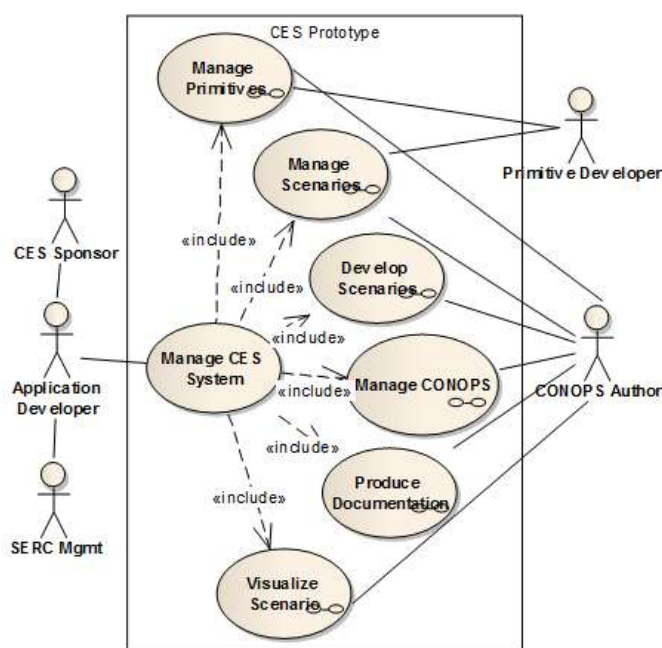


Figure 4: CES Prototype Top Level Use Case Diagram

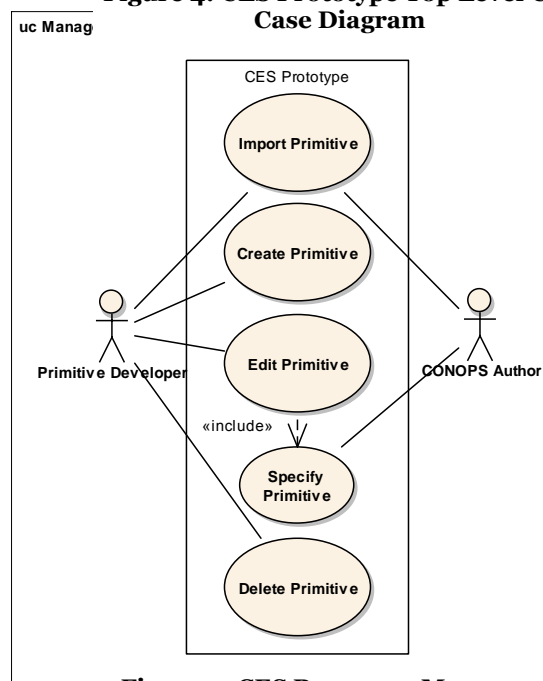


Figure 5: CES Prototype Manage Primitives Use Case

To guide development of the CES Prototype, models were developed following a Model-Based Systems Engineering (MBSE) methodology using the System Modeling Language (SysML). Use Case diagrams were developed to describe the nature of interaction between CES users and the CES system. From here, specific scenarios of possible use of CES were explored in Activity diagrams. Based on the Use Case and Activity diagrams, both black box and white box representations of CES were developed at a high level. These diagrams are shown below, representing desired CES Prototype functionality and architecture. As with all software development efforts, CES is constantly evolving and

updates to the CES model are made incrementally. These models are used as design tools for both development and analysis of CES programming efforts, allowing the research team to communicate with others and to reason with each other about the architecture of CES.

Use Case Diagrams

Figure 4 represents the high level interactions between CES and the CONOPS author (non-technical user) and primitive developer (advanced user). These interactions allow stakeholders and developers to create a graphical CONOPS model, produce a textual CONOPS artifact and create visualizations based on a CONOPS. Each Use Case is further elaborated in subsequent figures.

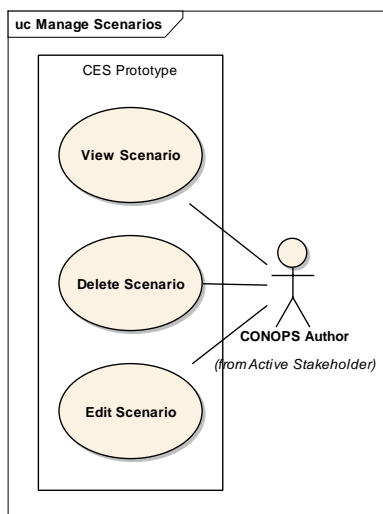


Figure 6: CES Prototype Manage Scenarios Use Case

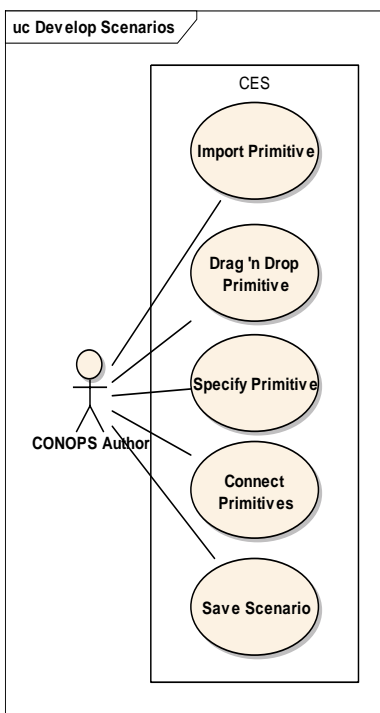


Figure 8: CES Prototype Develop Scenarios Use Case

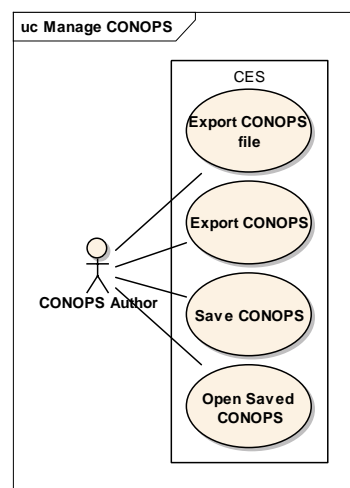


Figure 7: CES Prototype Manage CONOPS Use Case

In Figure 5, Manage Primitives describes the creation, use and modification of the basic building of a graphical CONOPS. Primitives are defined as the major elements or objects involved in scenarios built within the graphical CONOPS. They can represent people, locations, vehicles, /weapons, computer systems, etc. These primitives reside within a centralized database, allowing primitive developers to create them and make them available for use by CONOPS authors. Each primitive contains attributes that define the properties of the primitive, allowing CONOPS authors to alter primitive characteristics to their needs.

Scenarios, depicted in Figure 8 and Figure 7, involve the linking of individual primitives to tell the story of how a user may interact with a future system. By creating Scenarios, CONOPS authors will be able to describe their needs.

The combination of Scenarios created by the CONOPS author make up the graphical CONOPS model, which can be saved and manipulated in a format that

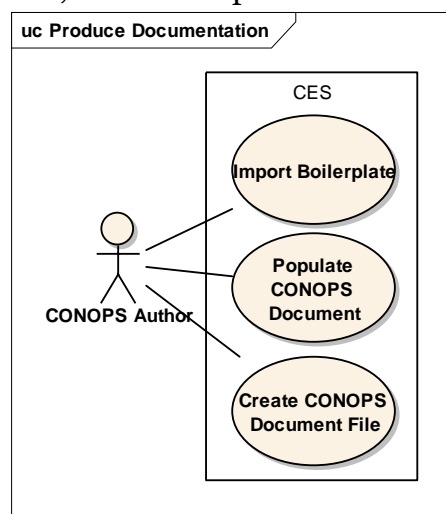


Figure 9: CES Prototype Produce Documentation Use Case

promotes reuse (Figure 6). Since a textual CONOPS is often a requirement for contracting purposes, CES will enable CONOPS authors to generate a textual CONOPS from a graphical CONOPS, as depicted in Figure 9.

Activity Diagrams

Activity diagrams give insight into the steps taken by both systems and their users to carry out use cases and provide capabilities to users. Activity diagrams are flow charts, displaying a set of activities carried out (rounded rectangles), by specific actors (vertical partitions) leading to the creation of certain artifacts (right angle rectangles). In Figure 10, the Manage Primitives Use Case described above is depicted in terms of the interaction between the CES Prototype, the Primitive Developer (advanced user) and an external 3D modeling software tool.

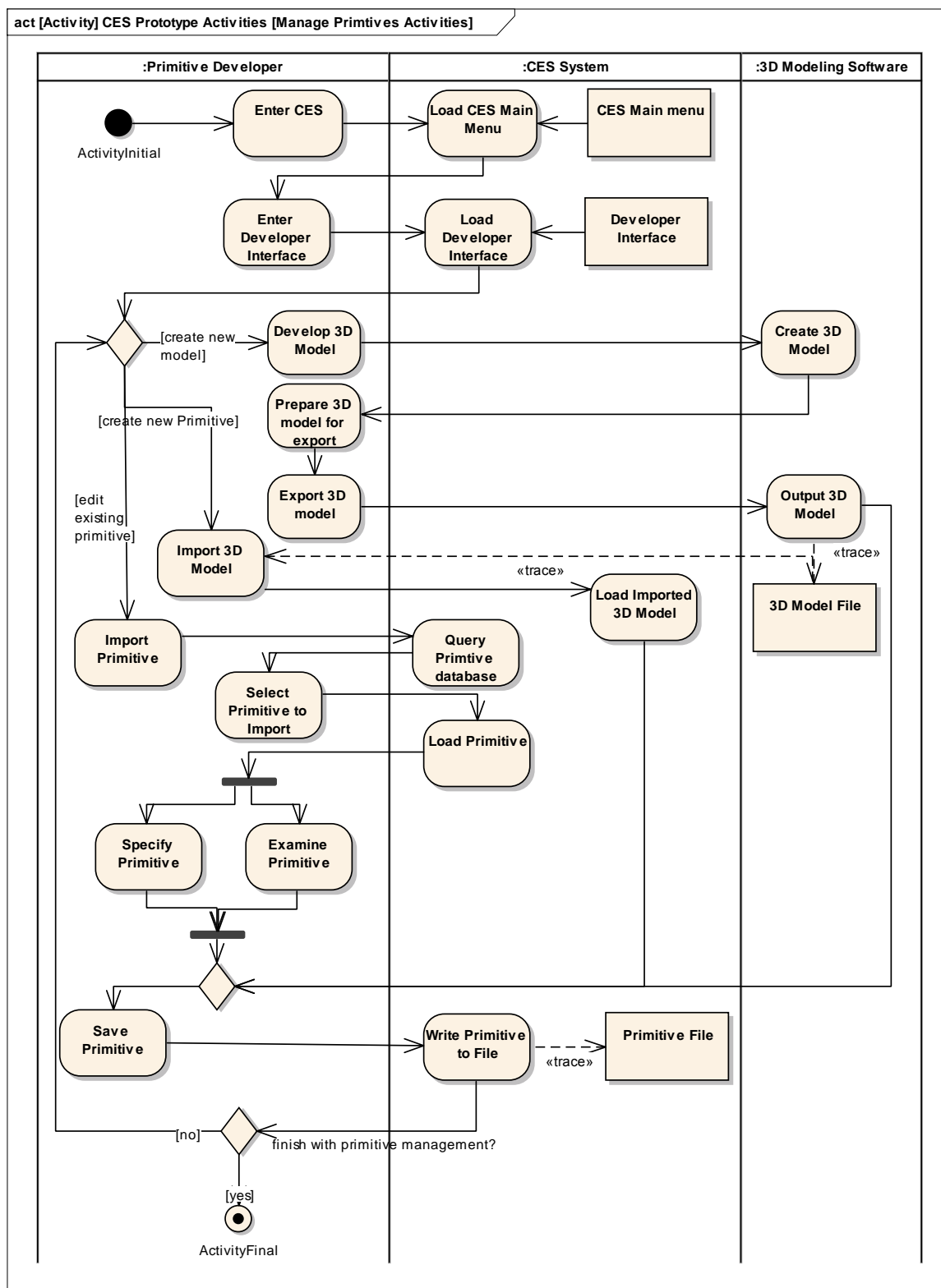


Figure 10: CES Prototype Manage Primitives Activities

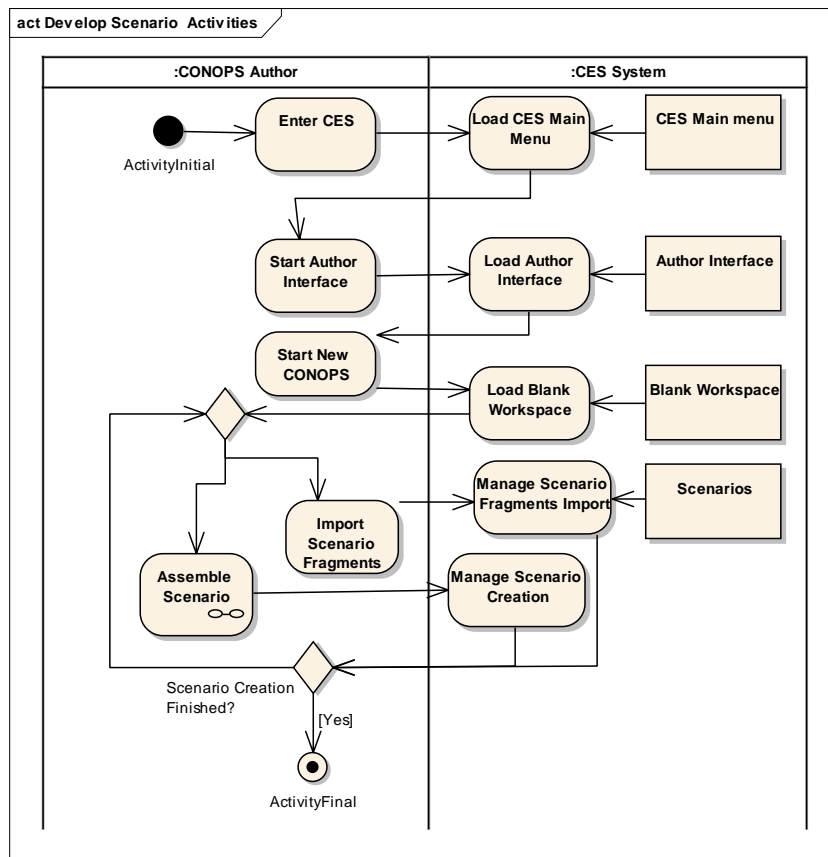


Figure 11: CES Prototype Develop Scenario Activities

Figure 11 represents typically activities required to carry out the Develop Scenario Use Case displayed in Figure 4. Each activity can be further explored with another Activity diagram, as is the case with Figure 12, which decomposes the Assemble Scenario activity, a task taken on by the CONOPS Author during the development of a scenario.

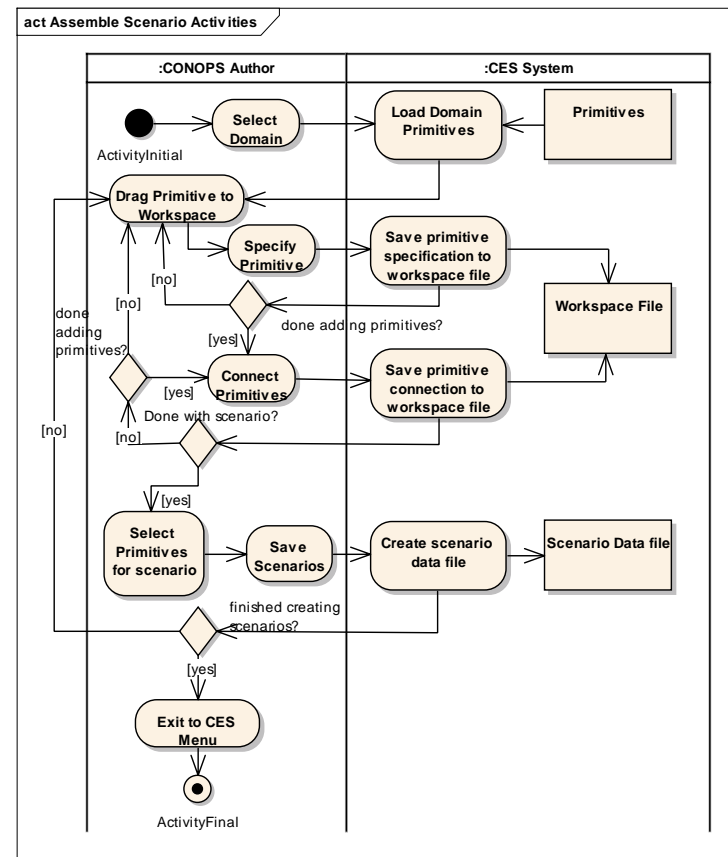


Figure 12: CES Prototype Assemble Scenario Activities

Architecture Models

Based on the Use Case and Activity diagrams, as well as the Conceptual Architecture depicted in Figure 3, a logical architecture for the CES prototype was developed. Figure 13 represents a hierarchical view of the major components CES will require, and how they can be decomposed.

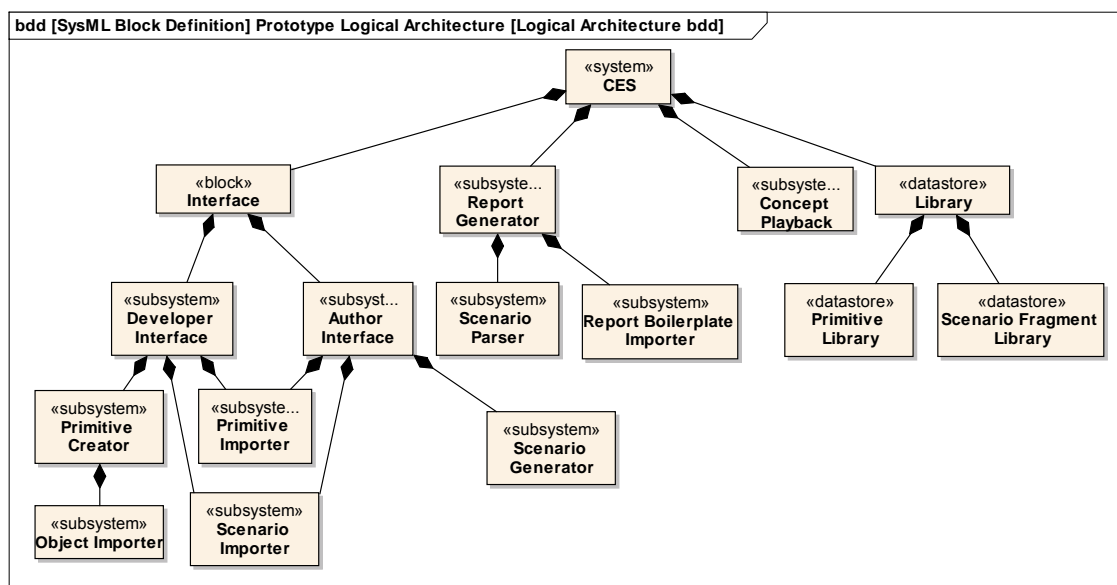


Figure 13: CES Prototype Logical Architecture

Figure 14 presents more detail relating to the interfaces between CES components. Each component is shown with ports, describing the flow of information and artifacts between components. These two logical architectures have helped guide the development of CES and will continue to be updated as the software architecture evolves.

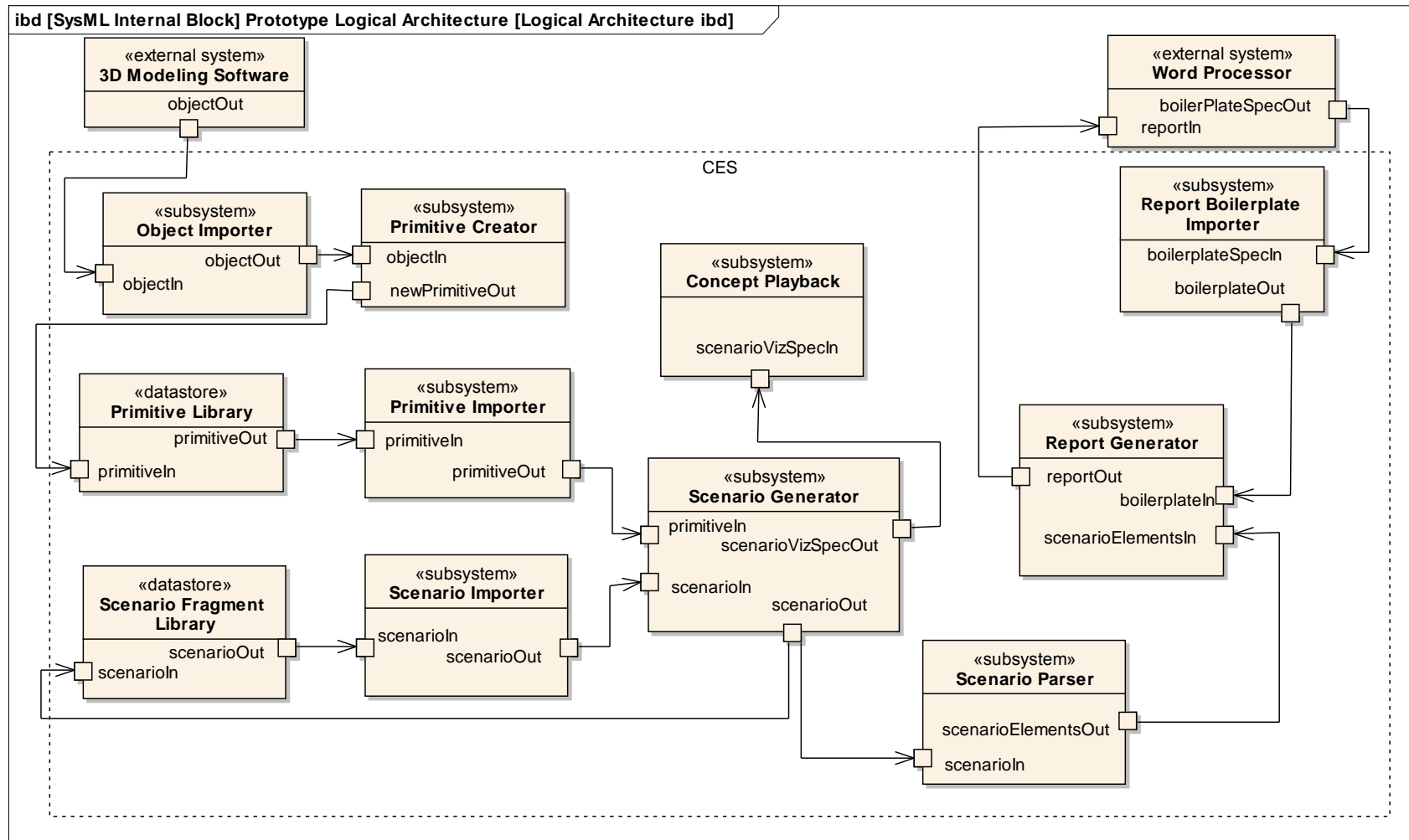


Figure 14: CES Prototype Logical Internal Architecture

3.2 PROTOTYPE FIRST RELEASE

Screenshots are provided below from the first release of the CES prototype. This prototype was provided to sponsor selected evaluators during a workshop, which will be discussed below.

Upon entering CES, the user is prompted to specify their role. The role the user will assume is based on their purpose in the CONOPS process and their technical expertise. Stakeholders and developers that are creating the CONOPS will enter the Author interface, while those supporting Authors through creation and specification of primitives, scenes and scenarios will enter in the Developer interface.

The Developer Interface, seen in Figure 16, provides an environment in which advanced users and system developers can create the materials CONOPS authors need to build the graphical CONOPS. This includes the creation or editing of primitives, the assignment of attributes to these primitives and the application of 3D representations to primitives.

When a user first enters the Developer Interface, they are welcomed by a help dialogue, providing them guidance. Whenever the user selects a command in the interface, a new help box is displayed. If the developer is an experienced user of CES, the help dialogue boxes can be automatically hidden, and recalled at any time.



Figure 15: CES Prototype Start Screen

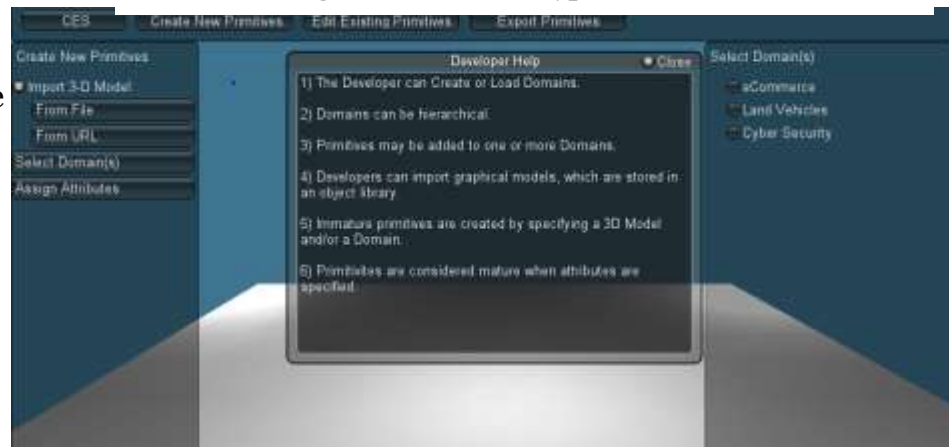


Figure 16: CES Prototype Developer Interface

This feature helps create a work flow for CES users, as well as providing an easy to assemble user manual.

The Author Interface is displayed in Figure 17, and includes 4 major areas. Dialogue box 1 represents the Author Interface help system, similar to Developer Interface. Area 2 controls the creation and specification of primitives and links, as depicted in Figure 18 and Figure 19. Area 3 acts as the graphical interactive workspace, where primitives can be added, linked, and moved. Finally, along the bottom and right sides of the screen, Area 4 provides controls to move between, create, delete and edit scenes. Each scene has a location, which is shown in the workspace (Area 3), and properties describing the transition between scenes (right hand side bar).



Figure 17: CES Prototype Author Interface

FIGURE 17 shows the author interface after the user has added four primitives, a man, a woman, a van and a car. From the primitive dialogue box on the upper left side, we can see that adding a primitive to the workspace creates an instantiation of that primitive, inheriting some of the properties of the parent primitive (car has speed profile, acceleration, fuel efficiency as attributes) but allowing the user to specify other properties (specific values for the attributes mentioned above, a specific brand, make, name, etc.).



Figure 18: CES Prototype: Adding Primitives

Figure 19 displays the Linking functionality of CES. Using the Link dialogue box, the user can specify the type of connection that exists between two primitives. In this scene, we can see that the female character Francie is recruiting the male character Bob and Bob is to drive the vehicle. While visible representations are not yet active on in the workspace, future development will allow some visual indicator of linkages. Figure 19 also shows an example pop-up where specific attributes may be entered relating to the link (for example where Bob is driving to, how fast, etc.).



Figure 19: CES Prototype: Connecting Primitives

So far, CES has assisted the CONOPS author in describing activities that the users and system will carry out in a specific increment of time. A goal of this research is to enable stakeholders to describe their interactions with a system during its operation. Therefore, a temporal component must be present in CES to allow representations of operational scenarios over time. Figure 20 shows that the Author Interface organizes scenarios into specific scenes, along the bottom of the screen. Figure 19 was the representation of scene 1; Figure 20 is the representation of scene 5. By allowing authors to create, reorder and move scenes, they are able to better tell the CONOPS story. A future goal of CES is to allow authors to describe the transitions between scenes in such a way as to allow auto-generation of an animation depicting the possible system/stakeholder interaction scenarios.



Figure 20: CES Prototype: Changing Scenes

3.3 USABILITY PLANNING

The team began to look at what might be involved with a complete usability approach for a virtual environment. That work is detailed in Appendix A. The questionnaires presented may be useful in soliciting information about the usability of the system. Any formal user testing will require approval the university Institutional Review Board (IRB) to ensure fair treatment of the test subjects.

4 WORKSHOP

To determine whether the CES development effort is applicable to the problems faced by the sponsor's workforce and can address the established research questions, a workshop was held on December 7, 2011. The workshop participants included: three members of the research team, two of the sponsor's SERC representatives and twelve representatives from the sponsor's organization. The representatives were from a number of areas within the sponsor's organization, in a variety of roles with a wide range of experience. Based on introductions made during the workshop, the following general biographic data was compiled

Table 3: Workshop Participant Data

Position	# Participants	# Years Experience	# Participants
Systems Engineer	11	0-5	2
"Concept Engineer"	5	6-15	3
Software Engineering	4	15-25	1
Tool Development	4	Over 25	7
Manager or Director	3		
System Architect	2		
Requirements Engineer	1		
Lead Systems Engineer	1		
Lead Software Engineer	1		
System Analyst	1		

Throughout the course of the workshop, it became clear that many of the participants were involved in concept development, CONOPS development, and establishing the capability baseline for system development. While there is no position currently called "Concept Engineer", the work that these participants were conducted can be classified as Concept Engineering, therefore the "Concept Engineer" label was added. The workshop participants were briefed on the research being conducted and introduced to the CES software. Basic capabilities of the software were described and the participants were asked to use CES to model scenarios developed by the research team. The scenarios are described below, followed by selected workshop feedback from participants.

4.1 SCENARIOS

During RT003 Phase 2, the News Agency Scenario was introduced (Cloutier et al., 2010). A taxonomy was developed for creating primitives that would be required to describe News Agency operational scenarios. In RT030, the News Agency taxonomy was utilized to develop a set of specific scenarios to be used for experimenting with and

validating the CES software. Four scenarios are displayed below using a single sentence and elaborated using a number of statements laying forth the actions required to carry out the scenario. An activity diagram also accompanies each scenario. The activity diagrams were developed as means to graphically represent the scenarios to enhance workshop participant understanding, as well as a tool for iterative design of the scenario.

4.1.1 NEWS AGENCY SCENARIO 1

A news agency deploying a reporter and support assets to a new story (Figure 21)

1. An anonymous email is sent to the NA claiming a major security breach took place in Bank resulting in the exposure of personal and financial information for thousands of customers.
2. NA assigns a reporter to verify the claim by the informant.
3. Reporter goes with crew and talks to the press office of Bank
4. Reporter A feels like he is not being told the truth
5. Reporter A conveys this to Editor, who agrees and sends out a support unit to start collecting footage for the evening edition
6. Support unit meets Reporter A and they record news stories to be played later
7. They transmit their footage to the Editor, who edits the video and prepares it for the evening edition

4.1.2 NEWS AGENCY SCENARIO 2

A reporter works to recruit a new contact for a story (Figure 22)

1. Editor needs to have the story confirmed before he can run it
2. Talks to his staff to see who might have contacts to push along the story
3. Reporter Rob says he knows someone at a security firm that handles the Bank's security business
4. Reporter Rob visits his contact, convinces him to share his knowledge and asks him questions
5. The source repeats a similar story to that heard at the Bank, with some inconsistencies.

4.1.3 NEWS AGENCY SCENARIO 3

A news agency assigning a reporter to independently corroborate a news source (Figure 23)

1. Before reporting on the story, the Managing Editor needs to corroborate the information gathered thus far.
2. Sends a Staff Reporter to the Fed oversight group
3. Staff Reporter gathers information from information bureau at the Fed oversight group
4. Staff Reporter returns to News Agency and confers with News Editor
5. News Editor evaluates corroboration, decides if it is sufficient for publishing
6. Managing Editor decides whether to run the story or not.

4.1.4 NEWS AGENCY SCENARIO 4

A news agency deploying a new reporter and support assets to follow-up on an existing story (Figure 24)

1. The News Agency receives reactions to a current or previous story.
2. The Editorial Board assesses the potential for a feature story.
3. The Editorial Team assigns Research Team to gather background information/previous stories
4. The Editorial Team assigns a new Reporter and crew to gather new information/interviews
5. A updated news story is created, including information previously collected along with fresh information

4.1.5 PRIMITIVE LIBRARY

Based on the News Agency taxonomy from RT003, along with the workshop scenarios described above, a set of entities or primitives that were required for the workshop was developed. This set includes any objects required to be able to model the News Agency Scenarios described above. Each object also contains the characteristics (attributes) of the object, as well as potential actions/activities (links) it can carry out. The object model for the initial CES Primitive Library can be seen in Figure 25.

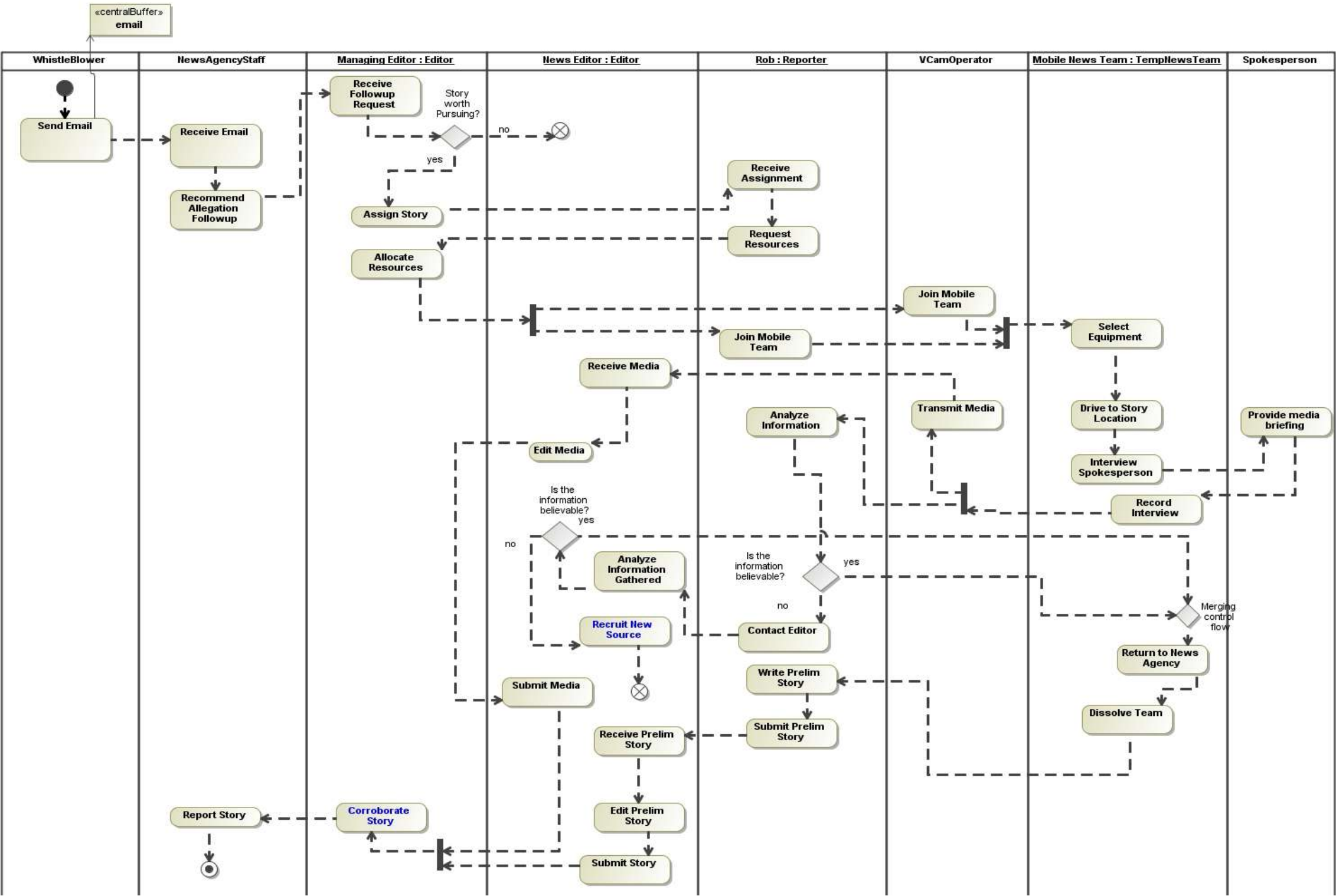


Figure 21: News Agency Scenario 1

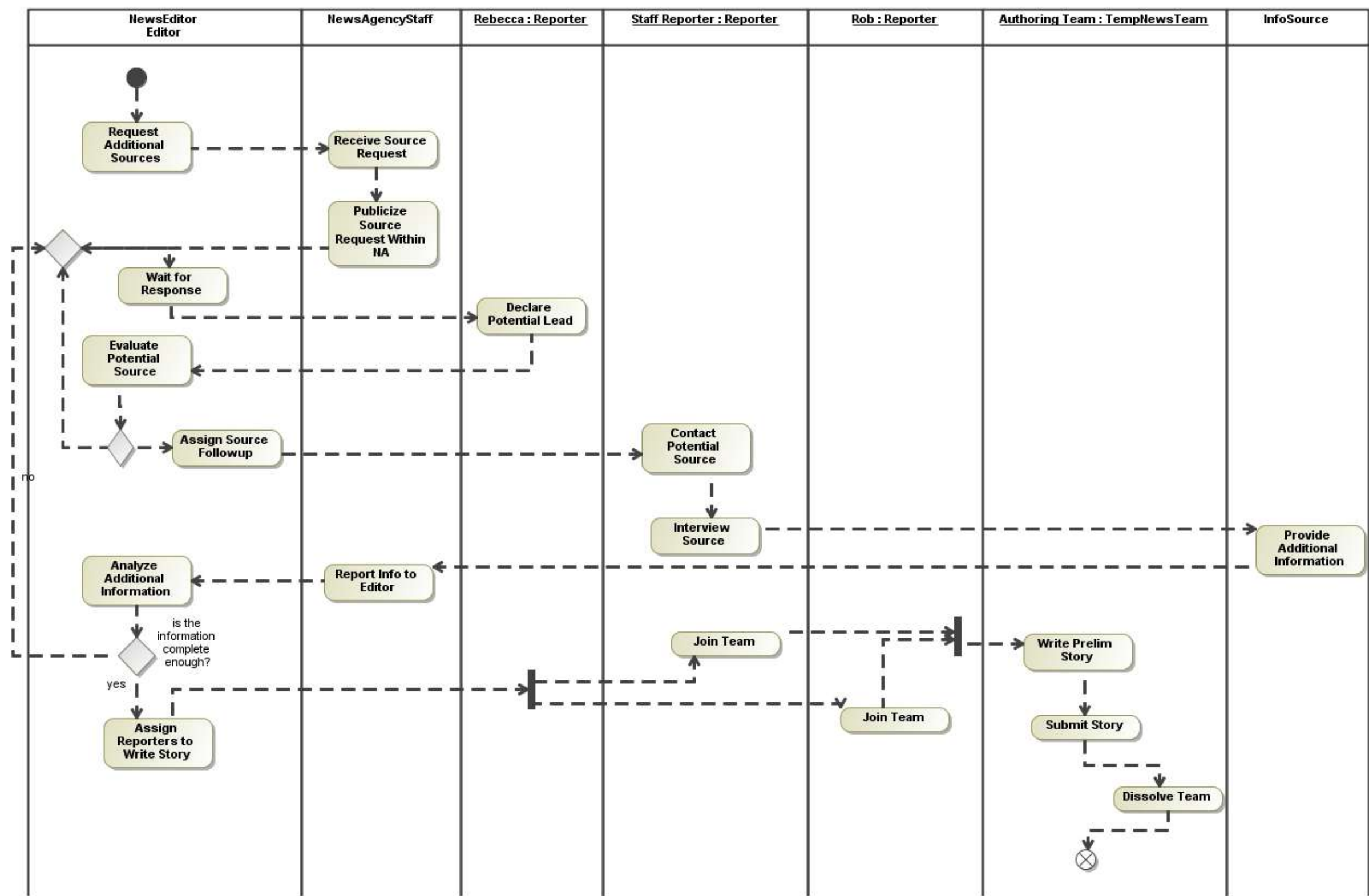


Figure 22: News Agency Scenario 2

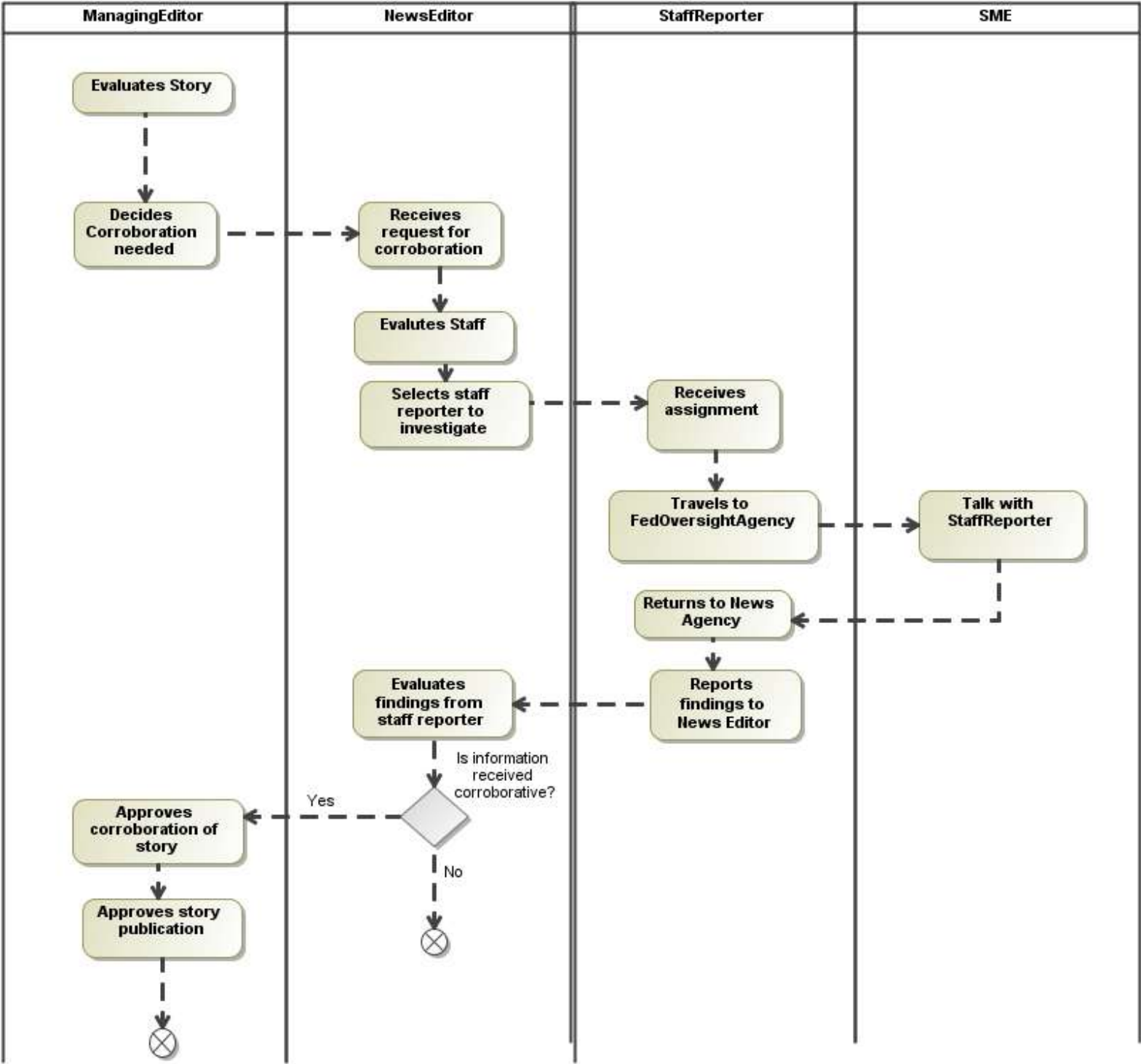


Figure 23: News Agency Scenario 3

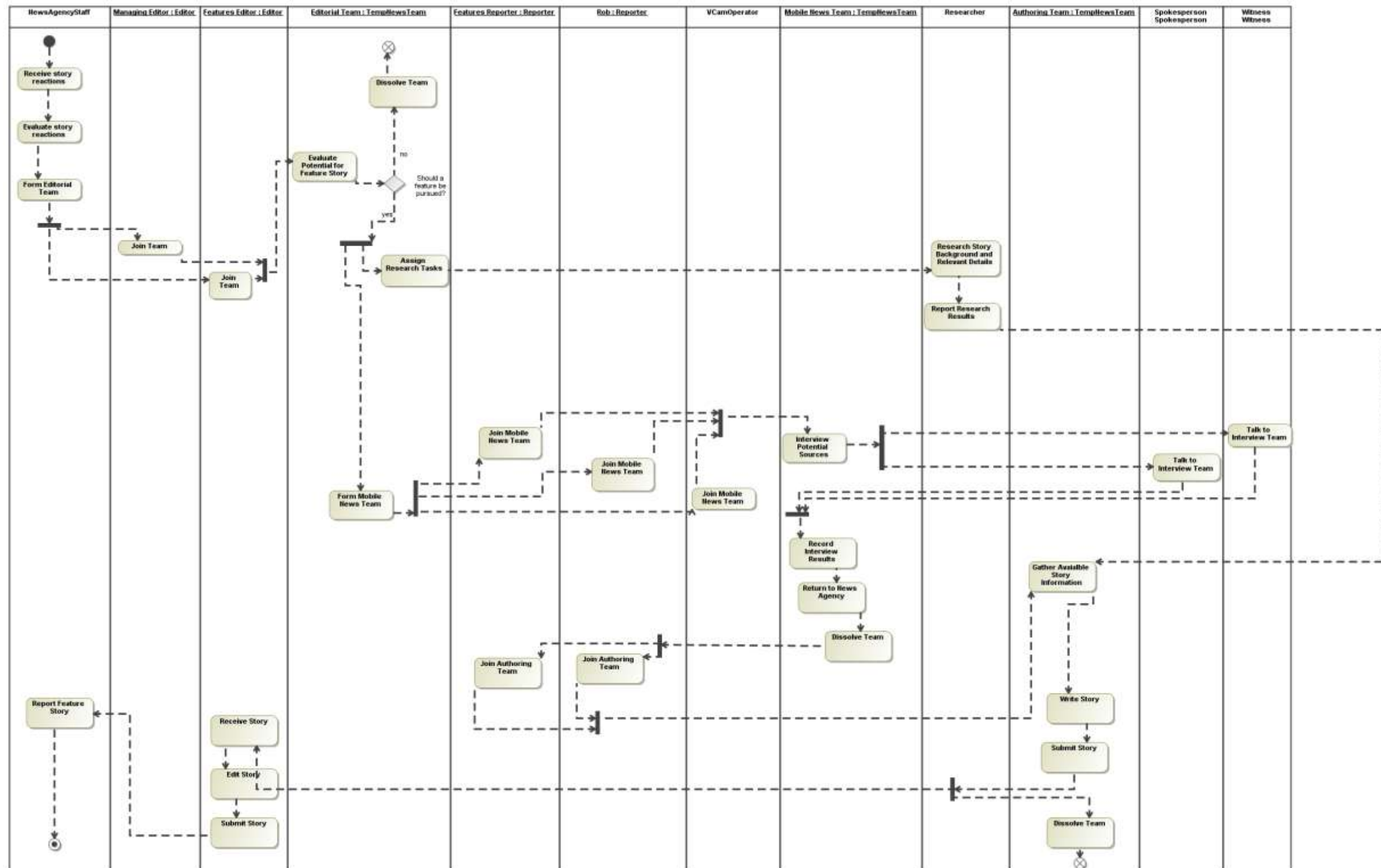


Figure 24: News Agency Scenario 4

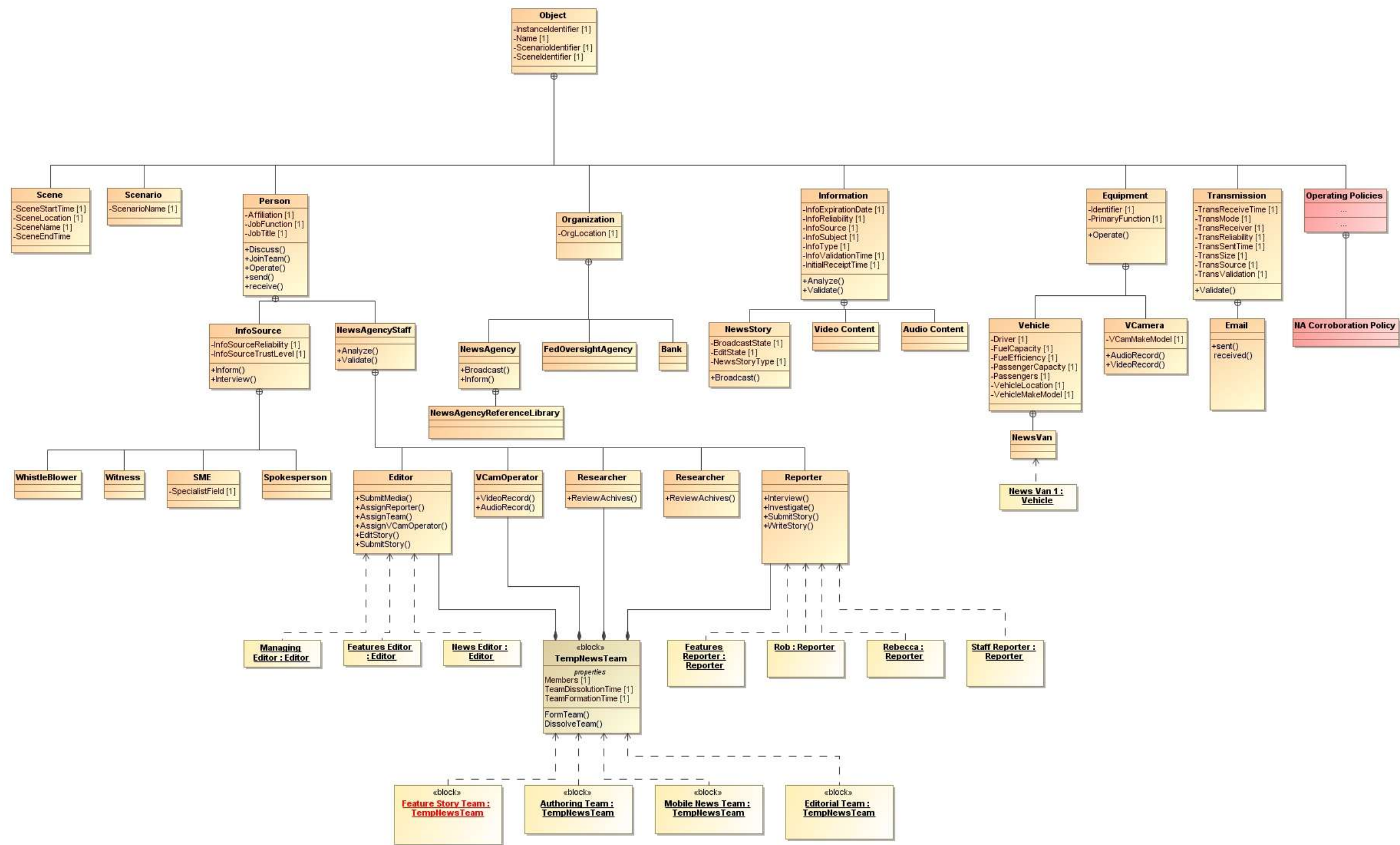


Figure 25: News Agency Scenario Primitives

4.2 WORKSHOP RESULTS

During the December workshop, participants were divided into four team four teams, with an average team size of four people each and each were provided a laptop and the CES prototype. The participants were provided Scenarios 1 and 2 as presented above. The collection of primitives programmed into the prototype was able to allow a full mapping of Scenario 1, while successfully representing Scenario 2 in CES would require some level of adaptation of the existing primitives and CES capabilities.

The groups were imaginative in their use of the software. Since it was a prototype, some of the functionality was limited and this led the users to push the software envelope quite a bit. Some of the observed activities included:

- re-purposing less mature existing 3-D objects and using them as placeholders in scenarios
- blurring the lines between what the tool allows users to do vs. what users want to do
- developing new scenarios, unrelated to the ones originally distributed for testing

4.2.1 EVALUATOR FEEDBACK

The feedback received from the workshop participants fell into three categories; positive aspects, detractors and suggestions for future development. Table 4 recounts some of the feedback collected:

Table 4: CES Evaluator Feedback

Positive
<ul style="list-style-type: none"> • the graphical representation and immersive environment led to a more realistic approach to CONOPS development • the idea of links and groupings was applicable for use in the evaluators' working environments • the immersive environment makes it easier to see anomalies and contradictions, and also makes it more enjoyable to visualize the scenario
Detractors
<ul style="list-style-type: none"> • difficulty was encountered specifying links between objects and "intangible objects" such as organizations • the software at its present level of immaturity did not lend itself to extension into other scenarios • there was no persistence of objects, since database was not implemented at this version

Suggestions/Observations

- add a notes section, to allow user annotation
- provide multiple camera views/perspectives
- graphically display links, rather than simply listing them
- filter links and objects, in order to sort and display only those elements of interest
- drag and drop objects from list
- provide “stubs” for objects, attributes and links, to enable authors to insert placeholders for missing or incomplete information
- provide animation for scene transition, if applicable
- physical space is limited, perhaps allow modelers to define their own terrains, maybe using real-world mapping interface
- develop new ontologies and taxonomies for different domains
- non-graphic concepts will need to be consistently represented in the environment

Feedback collected from workshop participants and sponsors was used to drive the goals and objectives of the next phase of this research, which will be discussed at the end of this report.

5 LESSONS LEARNED

There were many lessons learned during this phase of research. They are grouped by topics: Research, Project Management, System/Software Architecture, and Project/Code Construction.

5.1 RESEARCH LESSONS/QUESTIONS

1. The software effort is actually secondary to the research questions being asked, but consumes 98% of effort in the early stages – keep the research questions at the forefront of each member's attention.
 2. Can the process of CONOPS development and understanding be improved through the use of a graphical user interface?
 3. What's missing in a graphical scenario building paradigm?
 4. What's gained from the graphical scenario building paradigm?
 5. Does real-time collaboration between distributed stakeholders improve the CONOPS development?
 6. Can a real-time collaboration environment enable quicker consensus on CONOPS generation?
 7. Are there new or specific issues in asynchronous software development in an immersive environment?
-

5.2 PROJECT MANAGEMENT

1. It is critical to continuously monitor migration to new development environment releases – we now only migrate as a team, and then only after testing current builds in new release.
2. Iterative development tasks must be clearly defined/described to avoid redundant efforts and conflicts; this takes a lot of time and effort

3. Use of Agile processes very difficult in academia – neither students nor faculty are regular in their schedules/work times
4. Because of the above, the use of Skype and Google+ hangouts became very effective, and especially when it came to review, walk-through, testing, and team-effectiveness. Going forward, we would initiate daily meetings of short duration, to assess progress and discuss modifications
5. International composition of workforce has its own challenges
 - Clear Skype or Google+ hangouts, or even written word communication can be challenging - clarity can suffer when there's a lack of visual clues
 - Video conferencing is highly preferred over voice-only or written communication
 - Face-to-face remains the best way to manage, but video is a valuable 2nd best
 - Avoid idioms when describing scenarios and scenes
 - Analogies can work, but should be simple and clear
6. Measuring progress via visible functionality is not helpful, nor is using long-standing measures such as SLOC
 - Other criteria must be adopted and we propose a combination of SLOC count and a partitioning of categories of code: infrastructure, actual 3D object presentation, and 3D model manipulation
 - Current Statistics
 - SLOC for executable: 4,900, # objects: 199
 - SLOC for work in progress: 1,500, # objects: 47
7. Organization of code within the project listing is critical, especially when using a multiple-developer approach
 - Naming folders clearly is helpful
 - Grouping scripts together is critical, since most of the scripts are small (and again, naming clearly here is also critical to efficient searches)
 - Clarity among the team members is paramount, along with clear naming rules and conventions

8. The actual 3-D models used are free for academic and research use but are not free for commercial distribution; this is a consideration if deploying in an organization, so factor time for 3-D model development

5.3 SYSTEM/SOFTWARE ARCHITECTURE

1. Evaluation of scenarios was a key factor in an entire redraft and reconstruction of the architecture – representation of time (and activity ordering) became critical and factored into our final interface look and feel
2. All architecture and data objects should be specified as completely as possible and as early as possible
3. Because architecture (and infrastructure) is not “seen,” the work done is not obvious to management/ customer – and therefore it gives the impression of “no progress”
4. Error handling architecture should be context-reliant, and needs to be addressed for consistency
5. A subtle concept came to light – that is, the common formation of temporary teams for moving activities forward in a scenario. Having these “primitive groups” become objects with a life of their own requires new storage and naming facilities because the authors now, in effect, become developers – blurring the lines between the responsibilities and authorizations
6. In addition to the tracking of temporary groups in terms of data, considerations of representation also arose - these issues have been deferred to the next phrase; during the development of animation (or execution) of the scenes, temporary objects undergoing transformations (such as containment) will need a consistent strategy
7. We need to develop an ontological schema similar to semantic webs (such as OWL), and will research this moving forward
8. Representation of links is currently via a list, although it should also have a visible component, this was an issue because of potential graphical crowding - the use of filters was proposed and will be investigated moving forward

5.4 PROJECT/CODE CONSTRUCTION

1. Individual project access in asset server needs to be transparent to all developers
2. Managing Asset Server took more time than anticipated, and there was no easy way to roll-back to a previous release or version
3. There were occasional slow-downs when committing to or updating from the Asset Server
4. Highly-modular design vies with programming strategies – optimal breakpoints must be developed
5. Assignment of modular design elements is also problematic and, because of the iterative nature of design and development, is a real challenge
6. Graphic design of 3D models and manipulation of them, and scaling took longer than originally anticipated; this is not due to the provenance of the models, but is inherent to 3D environments
7. Avoid manipulation of the object surface meshes – in order to indicate a “selection” insert an indicator above the object itself
8. Along with time being a critical design component, the idea of simultaneous activity representation is also implicit in scenario building; we will investigate the accommodation of simultaneous activities in the next phase of design
9. An object drop default strategy is needed even if one is able to drag/drop objects – user-selected positioning is ideal
10. Movement of groups (for instance, a group of passengers entering a vehicle or building) will be crucial as the design moves forward, in terms of visual representation, as well as generated output
11. The movement of groups mentioned highlights another issue, that of containment of objects.

12. Manipulating colliders is how objects have solidity in the environment - this will be an important consideration when making scenes executable
13. Consider the use of multiple cameras as a default mechanism for each scene when being built, and provide an easy toggle for users to switch views

6 CONCLUSIONS

While these efforts are managed with a statement of work, and deliverables are identified, it must be remembered that it is research. The answers are not known, and the progress is unpredictable. To determine whether a graphical approach to CONOPS development is feasible and beneficial, we believe a tool must be developed. That work has progressed well. As detailed earlier in this report, the work performed during this research task included:

- Prototype architecture develop and modeled
- Prototype software developed and demonstrated
- Workshop conducted with sponsor selected personnel
- Refinement of research questions applicable to the research task
- Recommendations for future research and software development efforts for continuation of this research task are provided in the next section

The development approach attempted in this work did not work well, and that will have to be modified in the next phase.

The workshop conducted in December concluded with 16 participants, working in 5 groups, spent a couple of hours of “free-play” with the first iteration of the prototype. One notable item is that not a single group suffered a software crash. The underlying prototype infrastructure appears stable and robust.

The team is positioned well to begin a second iteration of the prototype, and to begin addressing the research questions identified. Lessons learned will be incorporated as the team moves forward.

7 RECOMMENDATIONS FOR CONTINUATION

Based on feedback from the workshop conducted during Phase 1, and meetings with research sponsors, a number of improvements to the CES prototype were suggested. The CES prototype will transform into the Integrated Concept Engineering System (ICES), which will contain third party simulation and analysis tool interfaces developed as part of RT0031. This will provide a better path to transitioning CES development, especially in the case of the “Animation” improvement discussed below, which must be driven by a simulation tool. The improvements below are seen as capabilities that could be added to the existing ICES prototype for the purpose of:

- progressing ICES development
- better fulfilling the needs of its users
- better positioning researchers to address RT0030’s research questions.

These improvements can be separated into those that require extensive programming (Major) and those that can be accomplished with tweaking of the existing prototype components (Minor). Minor improvements will definitely be programmed and completed during the next phase of research. Major improvements will be prioritized with the sponsor and are subject to change based on discussion with the sponsors.

Table 5: CES Prototype Minor Improvements

Minor Improvements (In rank order)	
Stubs	<ul style="list-style-type: none"> • Since the database of primitives is pre-populated, this task provides an opportunity to extend the current offerings by allowing the users to insert blank objects to the workspace. • This will help to maintain design flow without having to custom code full primitives • The presence of placeholders should generate a report to advanced users who can asynchronously code stubbed primitives into the database.
Link Visibility	<ul style="list-style-type: none"> • Currently, links between objects are listed textually by the prototype. • This improvement will create visible links between 3D models in the graphical workspace. • All links will be visible via a filter, the levels of filtering to be determined as development continues.
Annotation	<ul style="list-style-type: none"> • The user will be able to add annotation to objects. • This will allow users to denote status of updates/modifications, points of interest relevant to the scene and scenario, or simply to indicate outstanding questions to other users. • The annotation will be persistent (stored along with the current scenario in the database)
Multiple	<ul style="list-style-type: none"> • The users will be provided with a toggle for altering their view/perspective.

Views	
-------	--

Table 6: CES Prototype Major Improvements

Major Improvements (in rank order)	
Drag 'n drop	<ul style="list-style-type: none"> • Current prototype uses buttons to add primitives to the workspace at predetermined location and the objects cannot be repositioned. • User will be able to select objects via touch/mouse click and drag them to the workspace. • Position in the workspace can be varied via touch/mouse click and drag of primitives.
Animation	<ul style="list-style-type: none"> • Implement and enable a scene transition animation for motorized vehicles. • Underlying routines will access probability distributions to determine scene transition time, mimicking real-world phenomena. • This feature is connected to and dependent on external tool integration, discussed below.
External Tool Integration	<ul style="list-style-type: none"> • Prototype currently supports demonstrations of third party tool interfaces, • Additional development will extend and fully embed the use of such tools into the scenario building. • In relation to the Animation Work Task, the use of @Risk to determine time-to-destination is preferable to having a set time; it promotes a better modeling of real-world conditions. • Information reliability could also be assessed using mathematical tool/package, as would transportation or network flow optimization.
Database	<ul style="list-style-type: none"> • A pre-populated database will be created for the sponsor-defined scenarios. • Databases will include saving and retrieval of primitives, scenes and scenarios. • Will include functionality for users to save newly-created primitives and links to the database, for future retrieval.
Filtering	<ul style="list-style-type: none"> • User will have the ability to view various objects, links, or other scenario elements on their own, via a filtering capability. • As development continues, this philosophy of varying object and link visibility will be considered standard practice. • Results from this task will be used to assess the effectiveness of and effort required to using filtering to allow users to fully customize the environment
Split Screen	<ul style="list-style-type: none"> • The ability to view generated SysML representations of each scene will be explored. • The user will have the ability to split the screen and view the current scene in SysML mode. • The SysML representation will be updated (via refresh button) when changes are made to the graphical immersive environment

Additional research to be conducted during a future phase includes:

- *CONOPS development metrics* – To judge the effectiveness of ICES and the agile CONOPS development process, metrics must be developed to measure CONOPS development characteristics. Current literature will be reviewed and a number of metrics will be considered for use. Procedures will be developed to collect metric data during future workshops, including programming collection mechanisms into ICES where applicable.
- *Open Source Development* – Based on discussion with the sponsors, there has been some interest in examining the possibility of continuing ICES development using the Open Source paradigm. This Phase 2 will include investigation of existing open source development environments, and recommendations will be made.
- *Existing Taxonomies* – Based on feedback from the workshop, research will be conducted on existing taxonomies, ontologies and object libraries for their applicability to ICES development.
- *Detailed Architecture Modeling* – A continuous task in CES development has been modeling the ICES architecture at a high level. This high level modeling will continue, to be joined in Phase 2 with lower level architecture modeling. ICES components and interfaces will be modeled with more detail. This will allow the research team and sponsor to keep track of development efforts, keep the sponsor up to date on the design of ICES and help facilitate Technology Transfer with the sponsor at a technical level, if required.

APPENDICES

APPENDIX A - USABILITY

Usability is of critical importance for virtual environments, yet it is often one of the most neglected aspects of the development process. While several methods are available for testing the usability of virtual environment interfaces, no true consensus exists regarding which method works best in identifying problems in a user's experience that should be corrected. Additionally, each project is different, and the approach to usability evaluation needs to be tailored to account for such variances. Nevertheless, the goal of each evaluation is to improve the user's experience, and many of the same techniques can be applied, such as walkthroughs, user testing, and usability inspection, regardless of the application being reviewed.

A.1 INITIAL

The stage in the development process is critical when determining which evaluation method to use. For instance, competitive analysis is more useful in the early development stages when borrowing and generating ideas is needed, whereas user testing is more useful in the mid-to-late development stages when mockups have already been created. A general rule though is that evaluation is a continuous process that should account for a significant portion of the overall project cost. While the actual percentage will depend on the project, estimates range from a quarter of the overall budget (Brinck, 2002 p. 23) to ten percent (Nielsen, 2003). Nevertheless, when project budgets are being exhausted and costs need to be trimmed, usability evaluation often gets shorted. However, it is almost always better to reduce the scope of the project than spare usability of the product. Figure 1 illustrates the presence of usability specialists throughout all stages of the development process.

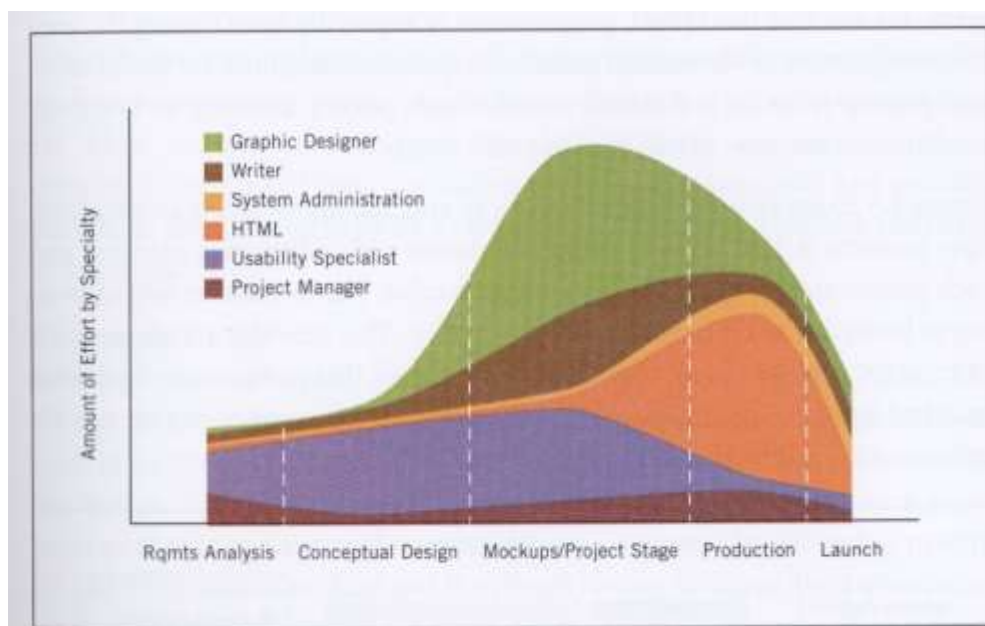


Figure 1: Staffing by Project Stage. The usability specialist is most involved during the initial stages of the development process, in order to identify problems as early as possible (Brinck, 2002).

Before conducting any usability evaluation, it is important to establish clear usability goals. For instance, the goal could be to mine the CONOPS virtual environment for existing problems, but the goal could also be criterion testing (does this site meet the explicit goals set for it?) or comparison testing (which of two or more designs is better?). The following recommended approach will primarily pertain to the first goal, or finding out what problems users may encounter.

Five common stages throughout the development of virtual environments are requirement analysis, conceptual design, mockups/prototypes, production, and launch and maintenance. Pervasive usability, or applying usability methods in every stage of the design process, is essential for identifying problems as soon as possible and saving time and money later on in the design process. The pervasive usability process is illustrated in Figure 2, and a comparison between early versus late-stage evaluation is shown in Table 1.

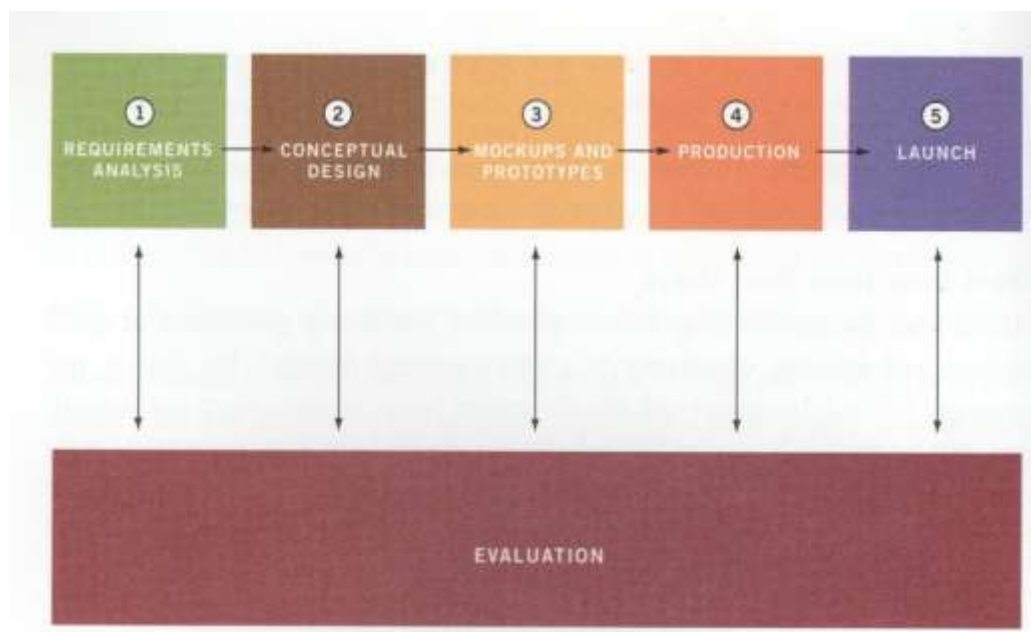


Figure 2: The Pervasive Usability Process (Brinck, 2002 p. 16).

	Early Evaluation / Testing	Late Evaluation / Testing
Advantages:	<ul style="list-style-type: none"> – Identify problems early, preventing costly redesign – Identifies major conceptual issues, such as problems with your overall organization and set of features 	<ul style="list-style-type: none"> – Finds specific problems – Finds problems throughout the site – Explores final task as closely as possible to how it will work when the site goes live
Disadvantages:	<ul style="list-style-type: none"> – Detailed problems are not possible to identify because the details haven't been specified yet – The user tasks are still somewhat artificial and their behavior is therefore somewhat contrived 	<ul style="list-style-type: none"> – When big problems are found, it may be too late to address them within the project budget

Table 1: Comparison of early evaluation testing in the design process with late evaluation testing (Brinck, 2002).

This project has already concluded the early development, and the corresponding assessment, under RT-03. We therefore focus herein on the mid-development and end-development processes required to test the prototype developed under RT-30 and the final product.

A.2 MID-DEVELOPMENT

Mockups/Prototypes

In the third stage of the development process, mockups (visual representations) and prototypes (interactive representations) for the final design are created and refined. Producing mockups early and rapidly, with efficient evaluation between each mockup, allows multiple opportunities to refine and elaborate on the design. This is crucial, because once production is undertaken a tremendous amount of time and money could be required for any significant changes occurring thereafter. Methods for evaluation during this stage include focus groups, walkthroughs, user testing, and usability checklists.

Focus groups are similar to interviews, but with the main difference being that focus groups entail gathering a group of people, often six to twelve, together to discuss the issues at hand instead of a single individual (Brinck, 2002). Focus groups can have problems associated with speaker bias and groupthink, but they can also delve into far greater detail on some issues than interviews. This is because people can develop ideas raised by one another and follow up lines of thought that the interviewer might not have known to pursue. Additionally, since groups are more difficult to coordinate, the facilitator may need to help manage interpersonal interactions and at times even foster arguments, as arguments can lead to a lot of information about why people feel the way they do. Due to this complex nature, professional facilitators or multiple facilitators may be needed. Three to five heterogeneous focus groups representative of the target audience is usually sufficient, as additional groups maybe not provide any substantial new information to justify the additional cost (Brinck, 2002). Focus groups can be used earlier in the development process, where the lessons learned will enable them to have a bigger impact on the final design, but conducting them when some mockups have already been created allows participants to react in a more specific and concrete way. An example of a focus group preparation worksheet is provided in Figure 3.

Focus Group Preparation Worksheet	
Project	_____
Dates and times	_____
Location	_____
Facilitator and other observers	_____
Required demographic	_____
Number of groups	_____
Number of people (per group)	_____
Payment (per person)	_____
Food and refreshments	_____
Videotaping and audiotaping:	video / audio / none
Recruiting ad	Where to place it? _____
	Wording _____
Questions to ask:	_____

Materials	
Check that you have each of the following, as needed, for your focus group.	
<input type="checkbox"/>	Consent form
<input type="checkbox"/>	Demographic questionnaire
<input type="checkbox"/>	Debriefing sheet
<input type="checkbox"/>	Mockups
<input type="checkbox"/>	Observer notes sheets
<input type="checkbox"/>	List of participants
<input type="checkbox"/>	Name tags
<input type="checkbox"/>	Payment checks
<input type="checkbox"/>	Audio and videotape
<input type="checkbox"/>	Seating chart

Figure 3: Focus Group Preparation Worksheet (Brinck, 2002 p. 90).

In addition to focus groups, walkthroughs can be used to improve usability by identifying labeling and placement problems early on. This is done by leading people through a mockup and asking them for their reactions as they go. Feedback on the look and concept of the virtual environment is easier to achieve through walkthroughs than through user testing, because the user will be less focused on problem-solving. Thus, the user is freed up to mention odd aspects of the interface, text they do not like, or their own design tastes.

As with focus groups and walkthroughs, user testing involves bringing in actual users to help in the evaluation of the virtual environment interface. Early user testing on mockups and prototypes, done well before the site is complete, helps examine how a user might react to a design. This will ultimately allow the site to be developed more quickly and cost-effectively, with an end result that is more user-centered and successful. Errors can be identified, specific design ideas can be tested for feasibility, and potential clients can begin to familiarize themselves with the interface (Brinck, 2002). A worksheet for user testing preparation is shown in Figure 4.

<p>User Testing Preparation Worksheet</p> <p>User Testing Project _____</p> <p>Scheduled Dates and Times _____</p> <p>Location: _____</p> <p><input type="checkbox"/> Usability lab <input type="checkbox"/> On-site testing</p> <p><input type="checkbox"/> Other: _____</p> <p>Testers _____</p> <p>Observers _____</p> <p>Required demographic _____</p> <p>Number of test users _____</p> <p>Payment (per person) _____</p> <p>Computer configuration(s):</p> <p>Hardware _____</p> <p>OS and version _____</p> <p>Browser and version _____</p> <p>Other software _____</p> <p>Videotaping and audiotaping: Video / Audio / None</p> <p>Other equipment:</p> <p><input type="checkbox"/> Stopwatch</p> <p><input type="checkbox"/> Server logs</p> <p><input type="checkbox"/> Event logs (mouse clicks, keystrokes)</p> <p><input type="checkbox"/> One-way mirror</p> <p><input type="checkbox"/> Other: _____</p> <p>Preparation</p> <p><input type="checkbox"/> Check when testing location and materials are prepared.</p> <p><input type="checkbox"/> Check when pilot testing has been completed.</p> <p><input type="checkbox"/> Check when sample results have been analyzed.</p>	<p>User Recruiting</p> <p>Check that each of the following has been prepared:</p> <p><input type="checkbox"/> Recruiting ad</p> <p><input type="checkbox"/> Recruiting sign-up sheet</p> <p><input type="checkbox"/> Recruiting qualifier questions</p> <p>Materials</p> <p>Check that you have each of the following, as needed, for your test session:</p> <p><input type="checkbox"/> Testing script</p> <p><input type="checkbox"/> Experimenter notes pages</p> <p><input type="checkbox"/> Consent form</p> <p><input type="checkbox"/> Instruction sheet</p> <p><input type="checkbox"/> Task questionnaires</p> <p><input type="checkbox"/> Post-questionnaire and demographics sheet</p> <p><input type="checkbox"/> Follow-up sheet</p> <p>Task list</p> <p>To clarify the goals of this test, list the primary tasks that you'll be testing. The testing script should provide the exact wording for each of these.</p> <ol style="list-style-type: none"> 1. _____ 2. _____ 3. _____ 4. _____ 5. _____ 6. _____ 7. _____ 8. _____ 9. _____ 10. _____
---	---

Figure 4: An example of a user testing preparation worksheet (Brinck, 2002 p. 424-425)

Mockups and prototypes for user testing can vary anywhere from fuzzy layouts of general page requirements, known as low-fidelity mockups, to fine-grained digital versions that are highly elaborate, known as high-fidelity mockups. Low-fidelity mockups are tested first, and are useful for discovering larger problems. High-fidelity mockups are usually reserved for the production stage, and are addressed in more detail in the next section of this report.

On some occasions, improvements to the virtual environment can be identified without bringing in actual users. To accomplish this, usability checklists are helpful in that they provide a set of guidelines to evaluate potential usability problems. The checklists can be either short or long and either general purpose or special purpose. Once a checklist has been selected, a designer or usability specialist performing the inspection judges how specific elements of a user interface, or virtual environment, conform to the items on the

list. Identified problems are then fixed in regard to their severity and how easy they are to correct. An example of a usability inspection that can be used as a mockup checklist and a writing guidelines checklist are shown in Figure 5 and Figure 6, respectively.

<p>Mockup Checklist</p> <p>Layout</p> <ul style="list-style-type: none"> <input type="checkbox"/> Simplicity, consistency, and focus. <input type="checkbox"/> Contrast, balance, and repetition. <input type="checkbox"/> Proximity, similarity, and good continuation. <input type="checkbox"/> Critical elements stand out. <input type="checkbox"/> Critical information appears toward top left of the page. <input type="checkbox"/> Works for printing and at a variety of window sizes (e.g., 520 pixel maximum width of your design). <input type="checkbox"/> Provides appropriate focal point, emphasis, and hierarchy of information. <p>Background Image</p> <ul style="list-style-type: none"> <input type="checkbox"/> Can be compressed to a reasonable size. <input type="checkbox"/> Aligns with the foreground images. <input type="checkbox"/> Will tile appropriately. <p>Navigation</p> <ul style="list-style-type: none"> <input type="checkbox"/> Navigation is scalable. <input type="checkbox"/> The most complex page can be developed using this framework. <input type="checkbox"/> Proper page titles and link labels have been used. <p>Text/Fonts</p> <ul style="list-style-type: none"> <input type="checkbox"/> The typeface matches the page style. <input type="checkbox"/> The number of typefaces is limited. <input type="checkbox"/> The use of typefaces, weights, and emphasis is limited. <input type="checkbox"/> HTML text is aliased (jaggy) and presented in the expected font. <input type="checkbox"/> Font size is flexible. <input type="checkbox"/> Text links are underlined. <input type="checkbox"/> Text links are different colors for visited and unvisited links. <input type="checkbox"/> Body text, titles, and labels are legible. <p>Images</p> <ul style="list-style-type: none"> <input type="checkbox"/> A consistent light source is used. <input type="checkbox"/> The compression of the mockup does not lose too much visual quality. <input type="checkbox"/> The images are used to support the content of the page. <p>Color</p> <ul style="list-style-type: none"> <input type="checkbox"/> Color is used appropriately (e.g., for grouping, pop-out effects, and so forth). <input type="checkbox"/> Color is appropriate for dark, light, and grayscale monitor settings. <input type="checkbox"/> Contrast is appropriate for dark, light, and grayscale monitor settings. <p>Client Requirements</p> <ul style="list-style-type: none"> <input type="checkbox"/> Required logos, fonts, and colors are included in the mockup. <input type="checkbox"/> Page titles, button labels, and link names are accurate. <input type="checkbox"/> Appropriate identifying images and marks are included. <input type="checkbox"/> The client address is correct.

Figure 5: Example of a Mockup (or Prototype) Checklist (Brinck, 2002 p. 231).

Writing Guidelines Checklist		
Content <ul style="list-style-type: none"> <input type="checkbox"/> The content of the web site provides value to the user. <input type="checkbox"/> The writing supports the reader's task. <input type="checkbox"/> The user is not required to read or navigate through irrelevant material to reach relevant material. <input type="checkbox"/> The text includes a call to action. <input type="checkbox"/> The reader interacts with the text as much as possible. <input type="checkbox"/> The information is accurate, authoritative, and up to date. <input type="checkbox"/> The information will be easy to maintain. <input type="checkbox"/> Items that need to be regularly updated have been documented. 		
Readability <ul style="list-style-type: none"> <input type="checkbox"/> The text is comprehensible and targeted at the right reading level. <input type="checkbox"/> Sentences are short, direct, concrete, and active. <input type="checkbox"/> New information is grounded in known information. <input type="checkbox"/> Text is in lay language, avoiding jargon, insider references, and obscure humor. 		
Legibility <ul style="list-style-type: none"> <input type="checkbox"/> The typeface is legible and the font size is sufficient. <input type="checkbox"/> Italics are avoided except at large sizes. <input type="checkbox"/> Boldface and all caps are only used for short pieces of text. (Boldface is preferred over all caps.) <input type="checkbox"/> Text has sufficient contrast with the background color and is not placed over a conflicting pattern. 		
Scannability <ul style="list-style-type: none"> <input type="checkbox"/> Emphasis is provided with appropriate headings, lead-ins, and pull quotes. <input type="checkbox"/> Opening sentences and paragraphs summarize the content. <input type="checkbox"/> Text is short, simple, and concise. <input type="checkbox"/> Text is specific and objective. <input type="checkbox"/> Text is broken into useful chunks and bulleted lists. 		
Orientation <ul style="list-style-type: none"> <input type="checkbox"/> Page titles provide useful branding clues. <input type="checkbox"/> Headings match the reader's goals. <input type="checkbox"/> Readers know where they are and what each page is about. 		
(Continued)		
Pagination <ul style="list-style-type: none"> <input type="checkbox"/> The text is divided between pages based on user tasks (i.e., pages are divided so users can skip portions irrelevant to them). <input type="checkbox"/> If scrolling is required, the user has appropriate cues within the text that more material is present, and horizontal rules are avoided. <input type="checkbox"/> Pages are self-explanatory; each page stands in its own. 		
Technique <ul style="list-style-type: none"> <input type="checkbox"/> Fundamentals are sound: grammar, spelling, capitalization, and punctuation. <input type="checkbox"/> Tone is natural and accessible. <input type="checkbox"/> Style is consistent. <input type="checkbox"/> Terminology is unambiguous. <input type="checkbox"/> Active sentences are used. 		
Links <ul style="list-style-type: none"> <input type="checkbox"/> Static text is never blue or underlined. <input type="checkbox"/> Text links are left in the default color. <input type="checkbox"/> Different types of links are distinguished graphically (e.g., audio clips vs. video clips). <input type="checkbox"/> Link text is descriptive and specific. <input type="checkbox"/> Email links explicitly show the email address. <input type="checkbox"/> Links don't cross punctuation or line breaks. 		
Forms <ul style="list-style-type: none"> <input type="checkbox"/> The order of steps through the form is clear. <input type="checkbox"/> Submit buttons are clearly labeled with descriptive text. <input type="checkbox"/> Reset buttons are avoided. <input type="checkbox"/> Required fields are clearly labeled. 		
Metadata <ul style="list-style-type: none"> <input type="checkbox"/> Metadata description and keywords are provided for each page. <input type="checkbox"/> Non-body text is specific and consistent: titles, ALT text, captions, headings, and buttons. 		
Writing for Screen Readers <ul style="list-style-type: none"> <input type="checkbox"/> Text is concise. <input type="checkbox"/> The top of the page contains meaningful, page-specific information. <input type="checkbox"/> Link names are self-explanatory. 		
Format <ul style="list-style-type: none"> <input type="checkbox"/> The layout does not depend on a specific typeface or font size. <input type="checkbox"/> The specified typeface works with all platforms. <input type="checkbox"/> The default font size of the browser is used. <input type="checkbox"/> Semantic tags, rather than format tags, are used whenever possible. <input type="checkbox"/> Text aligns with graphics on the page and with other text blocks. <input type="checkbox"/> All centered text on the page is centered around one axis of symmetry. <input type="checkbox"/> Related text doesn't appear in multiple columns. <input type="checkbox"/> Headings are closer to their body text than to other text on the screen. 		
Intellectual Property <ul style="list-style-type: none"> <input type="checkbox"/> The copyright notice is present and in the correct format. <input type="checkbox"/> Trademarks and service marks follow corporate standards. <input type="checkbox"/> Company branding is strictly adhered to. <input type="checkbox"/> No information is confidential or sensitive. 		
Standards <p>For each of the following, specify the standards this site will follow.</p> <ul style="list-style-type: none"> • Person: <input type="checkbox"/> 1st and 2nd person (recommended), or <input type="checkbox"/> 3rd person • Commas before the last item in a list: <input type="checkbox"/> "A, B, and C" (preferred), or <input type="checkbox"/> "A, B and C" • Punctuation within quotes: <input type="checkbox"/> traditional punctuation ("text"), or <input type="checkbox"/> logical punctuation ("text") (preferred) • Header alignment: <input type="checkbox"/> left, <input type="checkbox"/> center, or <input type="checkbox"/> right • Paragraph format: <input type="checkbox"/> double-space between paragraphs, or <input type="checkbox"/> indented opening of paragraphs • Common spelling, hyphenation, and capitalization conventions, e.g.: <input type="checkbox"/> Internet or <input type="checkbox"/> internet, <input type="checkbox"/> web site or <input type="checkbox"/> website, <input type="checkbox"/> email or <input type="checkbox"/> e-mail • Preferred file format for text files (e.g., plain text, RTF, Word, HTML) • Preferred file format for figures (e.g., EPS, Photoshop, TIFF, PICT, BMP, gif, jpeg) 		

Figure 6: Example of a Writing Guidelines Checklist (Brinck, 2002 p. 257-259).

Heuristic evaluations, created by Jakob Nielsen, are a specific type of usability inspection. Nielsen derived 10 guidelines from a factor analysis of 249 usability problems, and then demonstrated that the 10 guidelines will identify the vast majority of problems (Nielsen, 1994). Nielsen's methodology included judging how well 101 usability heuristics, across 11 earlier projects, explained each of 249 usability problems. Since participation in the 11 original projects was necessary in order to assess the degree to which the heuristics explained the usability problems, additional raters besides Nielsen were not available. To determine how well each of the 101 usability heuristics explained each of the 249 usability problems, the following scale was used:

- 0 = Does not explain the problem at all
- 1 = May superficially address some aspect of the problem
- 2 = Explains a small part of the problem, but there are major aspects of the problem that are not explained
- 3 = Explains a major part of the problem, but there are some aspects of the problem that are not explained
- 4 = Fairly complete explanation of why this is a usability problem, but there is still more to the problem than is explained by the heuristic
- 5 = Complete explanation of why this is a problem

In essence, focusing on 10 guidelines facilitates a more cost-effective and quicker approach to finding most usability problems. In Table 2, a commonly used set of 10 web guidelines, taken from *Usability from the Web*, is compared to Nielsen's 10 guidelines. Similarities are in bold typeface, with the matching term located on an equivalent row.

Ten Web Guidelines (Brinck, 2002)	Heuristic Evaluation (Nielsen, 1994)
Content and Scope	Minimize the user's memory load
Speed	Shortcuts
Navigation	Clearly Marked Exits
Appropriateness to task	Good error messages
Visual Design	Feedback
Compatibility	Speak the user's language
Simplicity	Simple and Natural Dialogue
Consistency and contrast	Consistency
Error Handling	Prevent Errors
Respect for the User	Help and Documentation

Table 2: Comparing Topics between two Usability Inspection Methods.

Multiple inspectors should be employed for usability inspections because different evaluators find a wide variety of different usability problems. A study by Nielsen showing this phenomenon is displayed in Figure 7 and Figure 8, while Figure 9 shows an analysis on the optimal number of evaluators.

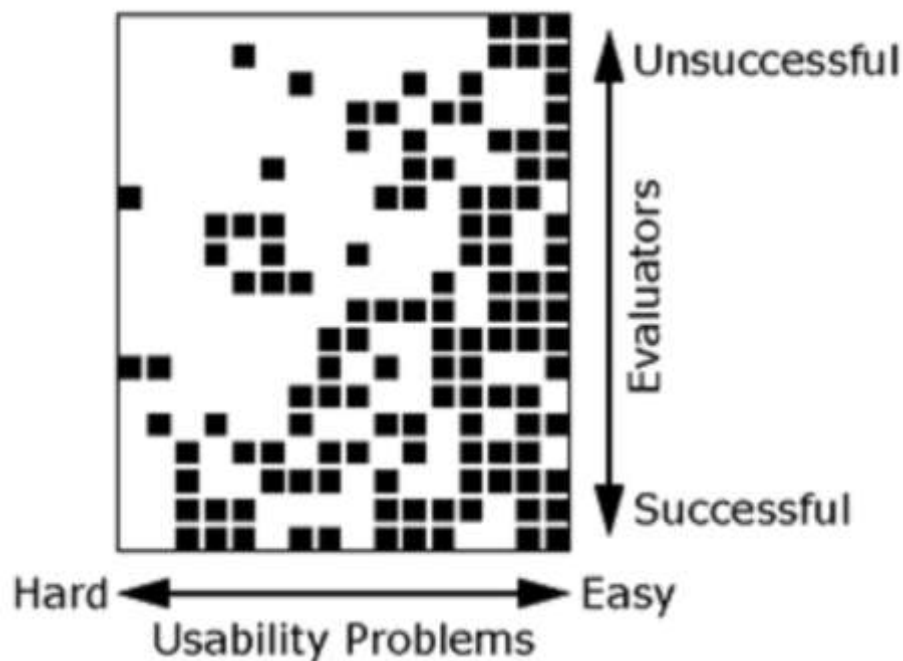


Figure 7: An example showing how different evaluators find different usability problems. Each black square represents a usability problem that was identified by the evaluator (Nielsen, 1994 p. 27).

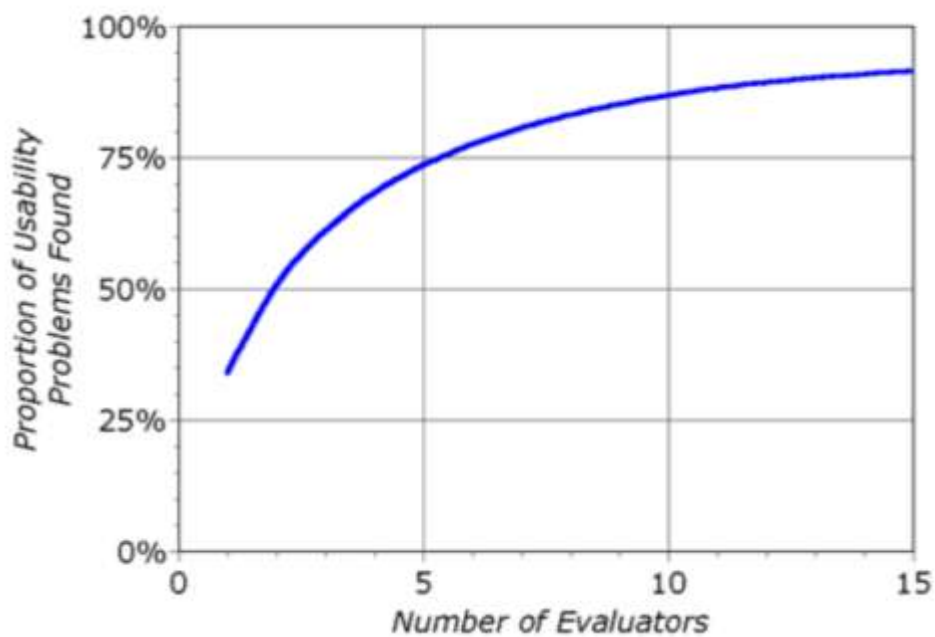


Figure 8: Curve showing the proportion of usability problems in an interface found by heuristic evaluation using various numbers of evaluators (Nielsen, 1994 p. 33).

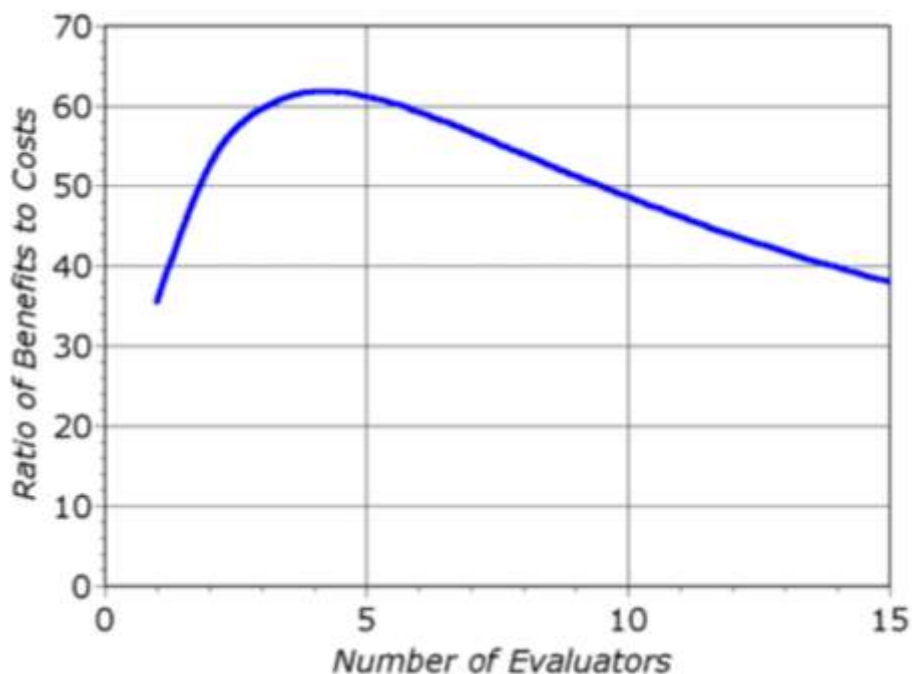


Figure 9: Curve showing how many times the benefits are greater than the costs for heuristic evaluation of a sample project using Nielsen's guidelines. The optimal number of evaluators in this example is four, with benefits that are 62 times greater than the costs (Nielsen, 1994 p. 35).

Walkthroughs, user testing, and usability checklists would ideally all be undertaken when evaluating virtual environment interfaces. However, budget constraints or time may not allow for the use of all three. Table 3 provides a comparison of the three methods. As can be seen, the three evaluation methods differ in cost, time to complete, level of expertise required to perform, and the type of usability problems often discovered. Thus, it is recommended that more than one method be used, or there will exist a much greater likelihood for usability problems to go unidentified. The size, scope, and stage of the project will determine which method is most appropriate.

The mockups and prototypes stage are an iterative process used to verify that the virtual environment has met the necessary design and usability goals. Improvements to the design are made with each additional iteration until the design is adequate for the next stage in the development process, which is production. A sample mockup development schedule is provided in Figure 10. The schedule is useful helping clients understand the consequences of making changes and requesting additional drafts beyond what was initially determiner. An example of a draft system model is shown in Figure 11. Clearly, this figures shows that the number of mockups produced at each draft stage decreases, and the more focused and refined the mockups should become. An example assessment sheet for a mockup style review is shown in Figure 12, with the draft number in the upper right corner serving as a gentle reminder that the iteration process cannot go on forever.

	Walkthroughs	User Testing	Usability Inspection / Heuristic Evaluation
Advantages	<p>Simple, Task-based</p> <p>Does not require actual system under analysis</p> <p>Can incorporate a wide variety of stakeholders, including management, customer support, sales or marketing representatives, and software developers</p> <p>More ideal than heuristic evaluations for generating design or interfaces</p> <p>Based on the language of goals, actions, and task breakdown, which has considerable face value in human-computer interaction</p>	<p>Relatively inexpensive</p> <p>Identifies extremely specific and practical problems</p> <p>High confidence in the results</p> <p>Users almost always find surprising problems that would not have been identified by other means</p> <p>Frequently, only a single tester is necessary</p> <p>User testing is possible to conduct even if the users do not know anything about user interface design</p>	<p>Relatively inexpensive</p> <p>Fast</p> <p>Can be completed alone</p> <p>Even a beginner can do a practical, useful inspection</p> <p>Generally more effective than walkthroughs when the two have been compared</p>
Disadvantages/ Limitations	<p>Can be expensive, as they may involve several people for significant periods of time</p> <p>Not as easy to learn or apply initially</p> <p>Learning the technique requires some exposure to psychological concepts and models of human problem solving and goal-oriented activity, with the procedure itself involving a systematic question and answer protocol</p> <p>Focus on just one attribute, easy of learning</p>	<p>Difficult and time-consuming to recruit participants that fit the ideal profile</p> <p>Should only use each user once</p> <p>Location of testing may be restricted</p> <p>Some users browse sites without explicit goals, making it extremely difficult to define appropriate tasks for testing</p> <p>Difficult to achieve statistically significant results at a low cost</p>	<p>Does not actually involve users in the evaluation process. Thus, it cannot be certain that identified problems reflect an actual problem for a user or that the solution will not be worse for the user</p> <p>Inspection problem reports typically do not predict end-user problems as well as one might wish, in order to confidently substitute inspection for end-user testing</p> <p>Inspections are not as effective in determining the overall satisfaction of customers as end-user testing</p> <p>Does not provide a systematic way to generate fixes to the usability problems or a way to assess the probable quality of any redesigns</p> <p>Often need multiple evaluators to find large proportions of problems</p>

Table 3: Comparing advantages and disadvantages/limitations of evaluation methods during the mid-development process.

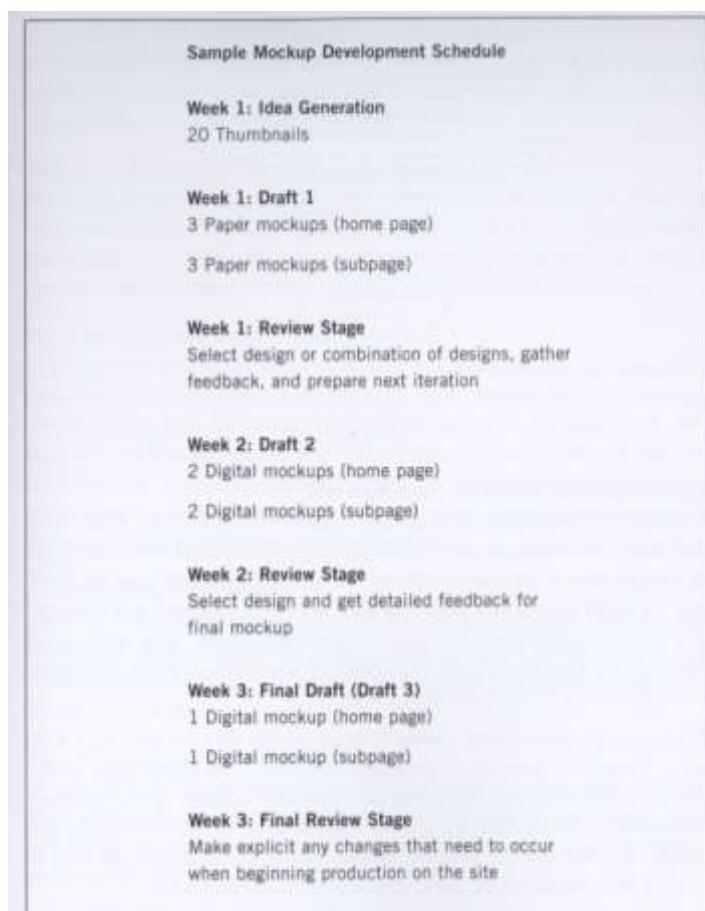


Figure 10: Example of a Mockup Development Schedule (Brinck, 2002 p. 233).

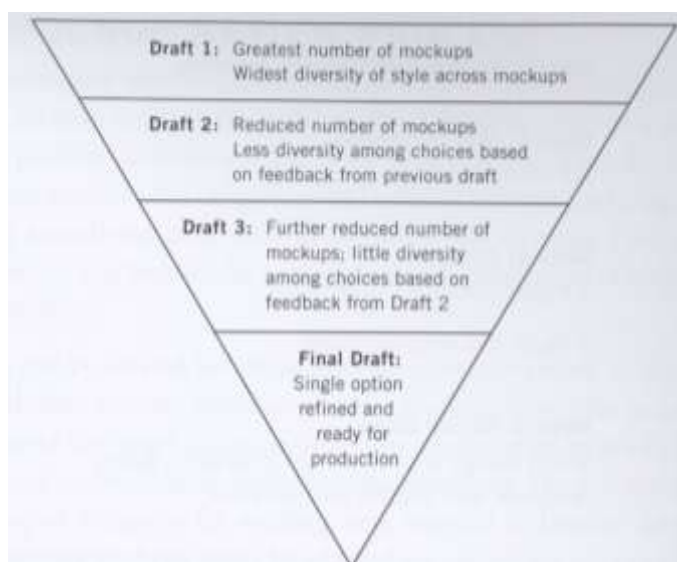


Figure 11: Example of a Mockup Development Schedule (Brinck, 2002 p. 234).

Mockup Style Review Form Date sent _____ Draft _____ of _____ Project _____ Project contact _____ Signing this form signifies the completion of the mockup style review process unless additional drafts have been previously arranged. Please return this completed form when you have finished reviewing the mockups to Company name: _____ Address: _____ Phone/fax numbers: _____ <input type="checkbox"/> Mockups are available for viewing online at (http://www.yourcompanyhere.com/clientname/mockups) <input type="checkbox"/> Attached as a printout are the following: <input type="checkbox"/> Home page <input type="checkbox"/> Subpage Mockup style to proceed with: 1 2 (3) (4) (5) (Other When Available): _____ Please review the following categories, and feel free to mix comments regarding multiple styles. Color scheme Do you want any color changes? <input type="checkbox"/> No <input type="checkbox"/> Yes If Yes, please explain: _____ <hr/> Size Do you have size constraints, limits, or specifications? <input type="checkbox"/> No <input type="checkbox"/> Yes If Yes, please explain: _____ <hr/> Navigation Do you have navigation or button title changes (refers to wording on the buttons or links)? <input type="checkbox"/> No <input type="checkbox"/> Yes If Yes, please explain: _____ <hr/> Is the navigation and button look acceptable? <input type="checkbox"/> No <input type="checkbox"/> Yes If No, please explain: _____ <hr/>		Are the general positioning and layout of navigation acceptable? <input type="checkbox"/> No <input type="checkbox"/> Yes If No, please explain: _____ <hr/> Graphics Are there any specific photographs or images you would prefer to see on the final web site? <input type="checkbox"/> No <input type="checkbox"/> Yes If Yes, please explain: _____ <hr/> Are company logos and other identifications included and correct? <input type="checkbox"/> No <input type="checkbox"/> Yes If No, please explain: _____ <hr/> General layout/look and feel Does the overall layout support your needs? <input type="checkbox"/> No <input type="checkbox"/> Yes If No, please explain: _____ <hr/> Does the look and feel match the image desired by your company? <input type="checkbox"/> No <input type="checkbox"/> Yes If No, please explain: _____ <hr/> Briefly describe the reason for preferring the mockup style you have chosen: _____ _____ _____ _____ <input type="checkbox"/> Additional comments are included with this form I have reviewed the final mockup style for this project and have determined that it is <input type="checkbox"/> Ready to proceed to the final draft <input type="checkbox"/> Ready to proceed to the next previously ordered draft <input type="checkbox"/> Not ready; I would like to purchase additional drafts Approved by: _____ Date of approval: _____ <hr/>
--	--	--

Figure 12: Example of a Mockup Style Review (Brinck, 2002 p. 236-237).

A.3 END DEVELOPMENT




Production

Once a mockup has been refined to the point where identified problems have been corrected, the final product is created. This includes coding the virtual environment with final text and graphic content. High-fidelity task-driven user testing, field testing, and quality assurance are useful methods of evaluation at this stage.

User testing in the production stage involves the use of high-fidelity mockups, which are more useful for refining the details of the design than the low-fidelity mockups used in earlier stages of the development process. For instance, user testing at this stage could involve assessing a variety of icon designs to find out which are most often identified successfully by the user. This can be done simply by showing the icons in context and asking the user to interpret what they think each icon means, and then determining how the worst icons can be improved. Two ways for conducting this user task are shown in Figure 13 and Figure 14. Such a process could be used for any graphics available in the virtual environment.

Form for Testing Whether an Icon Is Recognizable

Instructions: We've designed the following possible icons for a web site representing a business that wants to disseminate materials and research findings to the public sector. In the right-hand column, please write what each icon means to you.

Form for Testing Whether a Set of Icons Maps Uniquely to a Set of Concepts

Instructions: We've designed the following possible icons for a web site representing a business that wants to disseminate materials and research findings to the public sector. Please draw a line connecting each icon to the concept that it best represents.




	Late-Breaking Results
	Case Studies/White Papers
	Our Services

Figure 13 and Figure 14: Examples for Assessing Icon Usability (Brinck, 2002 p. 319-320).

Task-based quality testing is useful at this stage. The goal is to examine the processes that users will likely be interested in and then ensuring that they occur flawlessly. In other words, it is placing emphasis on exploring the most common and critical paths of the virtual environment. One method for accomplishing this is to draft a formal listing of the process tasks and results tasks that users should be able to achieve, and then testing to see if each of the processes can be achieved in a reasonable way. Problematic issues can be noted using a virtual environment feedback system to alert the design team. As every possible behavioral path through the virtual environment will not be tested, not all remaining problems will be highlighted. However, the formal listing of process and results tasks should identify any large issues that are remaining relating to the subset of tasks that were investigated, and serve as an alert for any major tasks that cannot be achieved. Figure 15 shows an example of a sheet that can be used for a task-based quality test.

Task or Process	Completed Task?	Comments
Can get from home page to contact page	√	No problems
Can successfully find the following book title: "My Goodness . . ."	√	Had slight trouble with initiating search
Can find the location of the library	√	
Can download a .PDF from the web site	No	Links broken to .PDFs! Need to fix; relative directory is incorrect
Readability and legibility sufficient (have read aloud)	√	No problems

Figure 15: A task-based testing sheet for a hypothetical library virtual environment (Brinck, 2002 p. 374).

Field testing takes user testing to the next step by testing the virtual environment in the actual domain in which it will be used, such as a workplace. This is useful for spotting a wide range of interaction problems such as software related issues and problems caused by the environment. Field tests are often expensive. Thus, to avoid suffering from

glaring usability problems that can be found more cheaply, it is recommended that they are supplemented with heuristic evaluation or laboratory-based user testing (Nielsen, 1994).

In addition to user testing and field testing, quality assurance testing is important throughout the development process to provide early design guidance and eliminate errors as early as possible. However, it reaches its pinnacle just before the virtual environment is launched. Even though a broken link or a misspelled word may seem like a trivial mistake, both can go a long way in undermining the credibility of the virtual environment. A solid quality assurance should rigorously examine editorial, graphics, and coding conventions. Checklists or a set of guidelines can be used to ensure the virtual environment is functioning properly. The level of assurance testing depends on the consequences of failure. For example, a virtual environment that provides a health-related service that could be life-saving would undergo more assurance testing than a simple online game. Additionally, a key concept of quality assurance is ongoing maintenance, so a provision for user feedback is often helpful.

Launch and Maintenance

Before launch, the final stage in the development process, a final quality testing phase is necessary to ensure that everything is ready. This phase could take as little as a few hours for a small site to upwards of a few weeks for a large site. After launch, correctness of the site should be verified immediately. In addition, the virtual environment should be maintained and refined, which involves accepting feedback and repeating the development process all the way from step one, requirement analysis.

Final quality assurance testing involves a more comprehensive testing of the virtual environment. It should ensure that the site will behave correctly in every instance and along every path the user may take. There should be no major errors or broken links. Possible forms of testing at this stage include a combination of code review, unit testing, automated testing, load testing, and outsourcing. The tests are summarized in Table 4.

Method	Definition
Code Review	Having another programmer (or several) read through the code line by line to verify that it implements the site's requirements and procedures.
Unit Testing	Taking a functional subset or unit of the system and verifying that the outputs are correct. Ensures that the proper inputs or parameters can produce relevant outputs.
Automated Testing	Write scripts or use automated link-testing programs to check all the links.
Load Testing	Test to ensure that a large number of simultaneous users or simultaneous transactions can be completed.
Outsourcing	Having specialized testing companies perform a wide variety of tests that are either too difficult or expensive to perform in-house.

Table 4: Definitions of Quality Assurance Tests (Brinck, 2002).

In addition to final quality assurance testing, some test completion criteria, or exit criteria, should be developed. In other words, this means that some specific goals need to be established and met in order to take the site live. If these goals are not met after testing, changes need to be implemented until a successful test of the completion criteria is achieved. Despite quality assurance testing, a few problems may still exist, particularly for a large site, and insisting on zero defects may lead to significant delays in the launch site. Thus, establishing a test criterion that allows for a small number of minor errors to remain may be necessary.

A.4 DISCUSSION AND RECOMMENDATIONS

Evaluating the usability of a design needs to be a continuous process throughout virtual environment development. Most evaluation methods are often not required elements of the design process. Yet, they are a key component of user-centered design, and therefore have a large impact in the development of a successful and highly-usable virtual environment. The primary reason for incorporating user feedback and testing as early as the prototype stage is to provide a concrete example of how a user might react to a design. By examining prototypes, users can reveal how well they understand the basic structure and whether they will find icons and labels straightforward and intuitive. A summary of usability methods and their role throughout virtual environment development is shown in Figure 16. The rest of this section is devoted to prototype evaluation recommendations as this is most relevant to the current phase of our project.

Usability Methods	Required?	Time to Perform	Costs	Learning Time	Confidence Level	Impact on Final Design
Requirements Analysis						
Requirements Specification	yes	short	—	short	low	high
User Interviews	no	medium	recruiting	medium	medium	high
User Surveys	no	long	mailings	medium-long	high	medium
Competitive Analysis	no	short	—	short	low	low
Conceptual Design						
Idea Generation	yes	short-medium	—	short	low	high
Information Architecture	yes	short-medium	—	medium	low	high
Card Sorting	no	short	recruiting	short	high	medium
Task Analysis	no	varies	—	medium	medium	high
Mockups and Prototypes						
Mockup Generation	yes	medium-long	—	long	low	high
Focus Groups	no	short	recruiting	medium	high	medium
User Testing	no	short	recruiting	medium	medium	medium
Production						
Usability Checklists	no	short	—	short	low	medium
User Testing	no	medium	recruiting	medium	high	medium
Launch						
Quality Assurance	yes	short-medium	patterns	medium	high	medium
Field Observations	no	long	varies	medium-long	medium	low
User Problem Tracking	yes	short	—	short	low	low

Figure 16: Framework for Contrasting Usability Methods (Brinck, 2002 p. 32-33).

Prototype Evaluation Recommendations

During the prototypes stage, many usability specialists, including Nielsen and Brinck, strongly recommend using both heuristic evaluation and user testing. The best approach for incorporating both methods is to iterate between the methods, because there is no reason to spend resources on evaluating an interface with many known usability problems only to have many of them come up again (Nielsen 1994). For instance, a heuristic evaluation would be performed first to clean up the interface and remove as many “obvious” usability problems as possible. Then, after a redesign of the interface, it would be subjected to user testing both to check the outcome of the iterative design step and to find remaining usability problems that were not picked up by the heuristic

evaluation (Nielsen 1994). Nielsen gives two major reasons for an iterative approach. First, a heuristic evaluation pass can eliminate a number of usability problems without the need to “waste users”, as users sometimes can be difficult to find and schedule in large numbers. Second, the two methods of usability assessment have been shown to find fairly distinct sets of usability problems, as opposed to repetitive findings, so they supplement each other well. After several iterations have been reviewed and redesigned, large discrepancies that may have existed will have ameliorated and a general consensus will emerge regarding the layout and usability.

In addition to the aforementioned rationale for an iterative approach, we offer two strategies for conducting iterative assessments of the virtual environment. First a small portion, or particular aspect, of the virtual environment can be examined during each iteration. Focusing on different aspects of the virtual environment allows for a detailed investigation that (1) does not require the entire design to be available and (2) ensures each aspect of the design is thoroughly examined. Second, for each subsequent iteration, we suggest increasing the amount of detail included in the checklist, as reviewing new elements will help uncover minor usability problems that may still exist in the interface.

Therefore, with the CONOPS virtual environment, we recommend the following steps iteratively to assess the basic structure and functionality:

1. Perform a usability inspection, such as a heuristic evaluation, on the prototype. See Figure 5 for a simple example checklist and the Appendix for a more complex checklist that may guide this process.
2. Redesign the interface to clean up any identified usability problems.
3. Perform user testing to check the outcome of the redesign and find remaining usability problems. See Figure 17 for a simple checklist that can facilitate this assessment.
4. Redesign the interface to clean up any identified usability problems.
5. Repeat steps 1 through 4 until the virtual environment has been refined to the point where it meets the final design specification and is ready for piloting.

Devoting extensive time and evaluation to prototypes greatly enhances the likelihood of getting the design right the first time around. Throughout the process, a wide range of alternatives can be explored with a reasonable cost of change. Repairing a design after the production stage has been reached will almost certainly be much more expensive.

<p>User Testing Preparation Worksheet</p> <p>User Testing Project _____</p> <p>Scheduled Dates and Times _____</p> <p>Location:</p> <p><input type="checkbox"/> Usability lab <input type="checkbox"/> On-site testing</p> <p><input type="checkbox"/> Other: _____</p> <p>Testers _____</p> <p>Observers _____</p> <p>Required demographic _____</p> <p>Number of test users _____</p> <p>Payment (per person) _____</p> <p>Computer configuration(s):</p> <p>Hardware _____</p> <p>OS and version _____</p> <p>Browser and version _____</p> <p>Other software _____</p> <p>Videotaping and audiotaping: Video / Audio / None</p> <p>Other equipment:</p> <p><input type="checkbox"/> Stopwatch</p> <p><input type="checkbox"/> Server logs</p> <p><input type="checkbox"/> Event logs (mouse clicks, keystrokes)</p> <p><input type="checkbox"/> One-way mirror</p> <p><input type="checkbox"/> Other: _____</p> <p>Preparation</p> <p><input type="checkbox"/> Check when testing location and materials are prepared.</p> <p><input type="checkbox"/> Check when pilot testing has been completed.</p> <p><input type="checkbox"/> Check when sample results have been analyzed.</p>	<p>User Recruiting</p> <p>Check that each of the following has been prepared:</p> <p><input type="checkbox"/> Recruiting ad</p> <p><input type="checkbox"/> Recruiting sign-up sheet</p> <p><input type="checkbox"/> Recruiting qualifier questions</p> <p>Materials</p> <p>Check that you have each of the following, as needed, for your test session:</p> <p><input type="checkbox"/> Testing script</p> <p><input type="checkbox"/> Experimenter notes pages</p> <p><input type="checkbox"/> Consent form</p> <p><input type="checkbox"/> Instruction sheet</p> <p><input type="checkbox"/> Task questionnaires</p> <p><input type="checkbox"/> Post-questionnaire and demographics sheet</p> <p><input type="checkbox"/> Follow-up sheet</p> <p>Task list</p> <p>To clarify the goals of this test, list the primary tasks that you'll be testing. The testing script should provide the exact wording for each of these.</p> <p>1. _____</p> <p>2. _____</p> <p>3. _____</p> <p>4. _____</p> <p>5. _____</p> <p>6. _____</p> <p>7. _____</p> <p>8. _____</p> <p>9. _____</p> <p>10. _____</p>
---	--

Figure 17: Example Checklist for User Testing (Brinck, 2002 p. 424-425).

A.5 HEURISTIC EVALUATION - A SYSTEM CHECKLIST

A.5.1 Visibility of System Status

The system should always keep user informed about what is going on, through appropriate feedback within reasonable time.

#	Review Checklist	Yes No N/A	Comments
1.1	Does every display begin with a title or header that describes screen contents?	O O O	
1.2	Is there a consistent icon design scheme and stylistic treatment across the system?	O O O	
1.3	Is a single, selected icon clearly visible when surrounded by unselected icons?	O O O	
1.4	Do menu instructions, prompts, and error messages appear in the same place(s) on each menu?	O O O	
1.5	In multipage data entry screens, is each page labeled to show its relation to others?	O O O	
1.6	If overwrite and insert mode are both available, is there a visible indication of which one the user is in?	O O O	
1.7	If pop-up windows are used to display error messages, do they allow the user to see the field in error?	O O O	
1.8	Is there some form of system feedback for every operator action?	O O O	
1.9	After the user completes an action (or group of actions), does the feedback indicate that the next group of actions can be started?	O O O	
1.10	Is there visual feedback in menus or dialog boxes about which choices are selectable?	O O O	
1.11	Is there visual feedback in menus or dialog boxes about which choice the cursor is on now?	O O O	
1.12	If multiple options can be selected in a menu or dialog box, is there visual feedback about which options are already selected?	O O O	
1.13	Is there visual feedback when objects are selected or moved?	O O O	

1.14	Is the current status of an icon clearly indicated?	O O O	
#	Review Checklist	Yes No N/A	Comments
1.15	Is there feedback when function keys are pressed?	O O O	
1.16	If there are observable delays (greater than fifteen seconds) in the system's response time, is the user kept informed of the system's progress?	O O O	
1.17	Are response times appropriate to the task?	O O O	
1.18	Typing, cursor motion, mouse selection: 50-1 50 milliseconds	O O O	
1.19	Simple, frequent tasks: less than 1 second	O O O	
1.20	Common tasks: 2-4 seconds	O O O	
1.21	Complex tasks: 8-12 seconds	O O O	
1.22	Are response times appropriate to the user's cognitive processing?	O O O	
1.23	Continuity of thinking is required and information must be remembered throughout several responses: less than two seconds.	O O O	
1.24	High levels of concentration aren't necessary and remembering information is not required: two to fifteen seconds.	O O O	
1.25	Is the menu-naming terminology consistent with the user's task domain?	O O O	
1.26	Does the system provide <i>visibility</i> : that is, by looking, can the user tell the state of the system and the alternatives for action?	O O O	
1.27	Do GUI menus make obvious which item has been selected?	O O O	
1.28	Do GUI menus make obvious whether deselection is possible?	O O O	
1.29	If users must navigate between multiple screens, does the system use context labels, menu maps, and place markers as navigational aids?	O O O	

A.5.2 Match Between System and the Real World

The system should speak the user's language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

#	Review Checklist	Yes No N/A	
2.1	Are icons concrete and familiar?	O O O	
2.2	Are menu choices ordered in the most logical way, given the user, the item names, and the task variables?	O O O	
2.3	If there is a natural sequence to menu choices, has it been used?	O O O	
2.4	Do related and interdependent fields appear on the same screen?	O O O	
2.5	If shape is used as a visual cue, does it match cultural conventions?	O O O	
2.6	Do the selected colors correspond to common expectations about color codes?	O O O	
2.7	When prompts imply a necessary action, are the words in the message consistent with that action?	O O O	
2.8	Do keystroke references in prompts match actual key names?	O O O	
2.9	On data entry screens, are tasks described in terminology familiar to users?	O O O	
2.10	Are field-level prompts provided for data entry screens?		
2.11	For question and answer interfaces, are questions stated in clear, simple language?	O O O	
2.12	Do menu choices fit logically into categories that have readily understood meanings?	O O O	
2.13	Are menu titles parallel grammatically?	O O O	
2.14	Does the command language employ user jargon and avoid computer jargon?	O O O	
2.15	Are command names specific rather than general?	O O O	
2.16	Does the command language allow both full names and abbreviations?	O O O	
2.17	Are input data codes meaningful?	O O O	

2.18	Have uncommon letter sequences been avoided whenever possible?	O O O	
2.19	Does the system automatically enter leading or trailing spaces to align decimal points?	O O O	
2.20	Does the system automatically enter a dollar sign and decimal for monetary entries?	O O O	

#	Review Checklist	Yes No N/A	Comments
2.21	Does the system automatically enter commas in numeric values greater than 9999?	O O O	
2.22	Do GUI menus offer activation: that is, make obvious how to say <i>"now do it"</i> ?	O O O	
2.23	Has the system been designed so that keys with similar names do not perform opposite (and potentially dangerous) actions?	O O O	
2.24	Are function keys labeled clearly and distinctively, even if this means breaking consistency rules?	O O O	

A.5.3 User Control and Freedom

Users should be free to select and sequence tasks (when appropriate), rather than having the system do this for them. Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Users should make their own decisions (with clear information) regarding the costs of exiting current work. The system should support undo and redo.

#	Review Checklist	Yes No N/A	Comments
3.1	If setting up windows is a low-frequency task, is it particularly easy to remember?	O O O	
3.2	In systems that use overlapping windows, is it easy for users to rearrange windows on the screen?	O O O	

3.3	In systems that use overlapping windows, is it easy for users to switch between windows?	0 0 0	
3.4	When a user's task is complete, does the system wait for a signal from the user before processing?	0 0 0	
3.5	Can users type-ahead in a system with many nested menus?	0 0 0	
3.6	Are users prompted to confirm commands that have drastic, destructive consequences?	0 0 0	
3.7	Is there an "undo" function at the level of a single action, a data entry, and a complete group of actions?	0 0 0	
3.8	Can users cancel out of operations in progress?	0 0 0	
3.9	Are character edits allowed in commands?	0 0 0	
3.10	Can users reduce data entry time by copying and modifying existing data?	0 0 0	
3.11	Are character edits allowed in data entry fields?	0 0 0	
3.12	If menu lists are long (more than seven items), can users select an item either by moving the cursor or by typing a mnemonic code?	0 0 0	
3.13	If the system uses a pointing device, do users have the option of either clicking on menu items or using a keyboard shortcut?	0 0 0	
3.14	Are menus broad (many items on a menu) rather than deep (many menu levels)?	0 0 0	
3.15	If the system has multiple menu levels, is there a mechanism that allows users to go back to previous menus?	0 0 0	
#	Review Checklist	Yes No N/A	Comments
3.16	If users can go back to a previous menu, can they change their earlier menu choice?	0 0 0	
3.17	Can users move forward and backward between fields or dialog box options?	0 0 0	
3.18	If the system has multipage data entry screens, can users move backward and forward among all the pages in the set?	0 0 0	
3.19	If the system uses a question and answer interface, can users go back to previous questions or skip forward to later questions?	0 0 0	

3.20	Do function keys that can cause serious consequences have an undo feature?	O O O	
3.21	Can users easily reverse their actions?	O O O	
3.22	If the system allows users to reverse their actions, is there a retracing mechanism to allow for multiple undos?	O O O	
3.23	Can users set their own system, session, file, and screen defaults?	O O O	

A.5.4. Consistency and Standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

#	Review Checklist	Yes No N/A	Comments
4.1	Have industry or company formatting standards been followed consistently in all screens within a system?	O O O	
4.2	Has a heavy use of all uppercase letters on a screen been avoided?	O O O	
4.3	Do abbreviations not include punctuation?	O O O	
4.4	Are integers right-justified and real numbers decimal-aligned?	O O O	
4.5	Are icons labeled?	O O O	
4.6	Are there no more than twelve to twenty icon types?	O O O	
4.7	Are there salient visual cues to identify the active window?	O O O	
4.8	Does each window have a title?	O O O	
4.9	Are vertical and horizontal scrolling possible in each window?	O O O	
4.10	Does the menu structure match the task structure?	O O O	
4.11	Have industry or company standards been established for menu design, and are they applied consistently on all menu screens in the system?	O O O	

4.12	Are menu choice lists presented vertically?	0 0 0	
4.13	If "exit" is a menu choice, does it always appear at the bottom of the list?	0 0 0	
4.14	Are menu titles either centered or left-justified?	0 0 0	
4.15	Are menu items left-justified, with the item number or mnemonic preceding the name?	0 0 0	
4.16	Do embedded field-level prompts appear to the right of the field label?	0 0 0	
4.17	Do on-line instructions appear in a consistent location across screens?	0 0 0	
4.18	Are field labels and fields distinguished typographically?	0 0 0	
4.19	Are field labels consistent from one data entry screen to another?	0 0 0	
4.20	Are fields and labels left-justified for alpha lists and right-justified for numeric lists?	0 0 0	
#	Review Checklist	Yes No N/A	Comments
4.21	Do field labels appear to the left of single fields and above list fields?	0 0 0	
4.22	Are attention-getting techniques used with care?	0 0 0	
4.23	Intensity: two levels only	0 0 0	
4.24	Size: up to four sizes	0 0 0	
4.25	Font: up to three	0 0 0	
4.26	Blink: two to four hertz	0 0 0	
4.27	Color: up to four (additional colors for occasional use only)	0 0 0	
4.28	Sound: soft tones for regular positive feedback, harsh for rare critical conditions	0 0 0	
4.29	Are attention-getting techniques used only for exceptional conditions or for time-dependent information?	0 0 0	
4.30	Are there no more than four to seven colors, and are they far apart	0 0 0	

	along the visible spectrum?		
4.31	Is a legend provided if color codes are numerous or not obvious in meaning?	0 0 0	
4.32	Have pairings of high-chroma, spectrally extreme colors been avoided?	0 0 0	
4.33	Are saturated blues avoided for text or other small, thin line symbols?	0 0 0	
4.34	Is the most important information placed at the beginning of the prompt?	0 0 0	
4.35	Are user actions named consistently across all prompts in the system?	0 0 0	
4.36	Are system objects named consistently across all prompts in the system?	0 0 0	
4.37	Do field-level prompts provide more information than a restatement of the field name?	0 0 0	
4.38	For question and answer interfaces, are the valid inputs for a question listed?	0 0 0	
4.39	Are menu choice names consistent, both within each menu and across the system, in grammatical style and terminology?	0 0 0	
4.40	Does the structure of menu choice names match their corresponding menu titles?	0 0 0	
4.41	Are commands used the same way, and do they mean the same thing, in all parts of the system?	0 0 0	
4.42	Does the command language have a consistent, natural, and mnemonic syntax?	0 0 0	
4.43	Do abbreviations follow a simple primary rule and, if necessary, a simple secondary rule for abbreviations that otherwise would be duplicates?	0 0 0	
#	Review Checklist	Yes No N/A	Comments
4.44	Is the secondary rule used only when necessary?	0 0 0	
4.45	Are abbreviated words all the same length?	0 0 0	

4.46	Is the structure of a data entry value consistent from screen to screen?	0 0 0	
4.47	Is the method for moving the cursor to the next or previous field consistent throughout the system?	0 0 0	
4.48	If the system has multipage data entry screens, do all pages have the same title?	0 0 0	
4.49	If the system has multipage data entry screens, does each page have a sequential page number?	0 0 0	
4.50	Does the system follow industry or company standards for function key assignments?	0 0 0	
4.51	Are high-value, high-chroma colors used to attract attention?	0 0 0	

A.5.5. Help Users Recognize, Diagnose, and Recover From Errors

Error messages should be expressed in plain language (NO CODES).

#	Review Checklist	Yes No N/A	Comments
5.1	Is sound used to signal an error?	0 0 0	
5.2	Are prompts stated constructively, without overt or implied criticism of the user?	0 0 0	
5.3	Do prompts imply that the user is in control?	0 0 0	
5.4	Are prompts brief and unambiguous.	0 0 0	
5.5	Are error messages worded so that the system, not the user, takes the blame?	0 0 0	
5.6	If humorous error messages are used, are they appropriate and inoffensive to the user population?	0 0 0	
5.7	Are error messages grammatically correct?	0 0 0	
5.8	Do error messages avoid the use of exclamation points?	0 0 0	
5.9	Do error messages avoid the use of violent or hostile words?	0 0 0	
5.10	Do error messages avoid an anthropomorphic tone?	0 0 0	

5.11	Do all error messages in the system use consistent grammatical style, form, terminology, and abbreviations?	O O O	
5.12	Do messages place users in control of the system?	O O O	
5.13	Does the command language use normal action-object syntax?	O O O	
5.14	Does the command language avoid arbitrary, non-English use of punctuation, except for symbols that users already know?	O O O	
5.15	If an error is detected in a data entry field, does the system place the cursor in that field or highlight the error?	O O O	
5.16	Do error messages inform the user of the error's severity?	O O O	
5.17	Do error messages suggest the cause of the problem?	O O O	
5.18	Do error messages provide appropriate semantic information?	O O O	
5.19	Do error messages provide appropriate syntactic information?	O O O	
5.20	Do error messages indicate what action the user needs to take to correct the error?	O O O	
5.21	If the system supports both novice and expert users, are multiple levels of error-message detail available?	O O O	

A.5.6. Error Prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place.

#	Review Checklist	Yes No N/A	Comments
6.1	If the database includes groups of data, can users enter more than one group on a single screen?	O O O	
6.2	Have dots or underscores been used to indicate field length?	O O O	
6.3	Is the menu choice name on a higher-level menu used as the menu title of the lower-level menu?	O O O	

6.4	Are menu choices logical, distinctive, and mutually exclusive?	0 0 0	
6.5	Are data inputs case-blind whenever possible?	0 0 0	
6.6	If the system displays multiple windows, is navigation between windows simple and visible?	0 0 0	
6.7	Are the function keys that can cause the most serious consequences in hard-to-reach positions?	0 0 0	
6.8	Are the function keys that can cause the most serious consequences located far away from low-consequence and high-use keys?	0 0 0	
6.9	Has the use of qualifier keys been minimized?	0 0 0	
6.10	If the system uses qualifier keys, are they used consistently throughout the system?	0 0 0	
6.11	Does the system prevent users from making errors whenever possible?	0 0 0	
6.12	Does the system warn users if they are about to make a potentially serious error?	0 0 0	
6.13	Does the system intelligently interpret variations in user commands?	0 0 0	
6.14	Do data entry screens and dialog boxes indicate the number of character spaces available in a field?	0 0 0	
6.15	Do fields in data entry screens and dialog boxes contain default values when appropriate?	0 0 0	

A.5.7. Recognition Rather Than Recall

Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

#	Review Checklist	Yes No N/A	Comments
7.1	For question and answer interfaces, are visual cues and white space used to distinguish questions, prompts, instructions, and user input?	0 0 0	

7.2	Does the data display start in the upper-left corner of the screen?	0 0 0	
7.3	Are multiword field labels placed horizontally (not stacked vertically)?	0 0 0	
7.4	Are all data a user needs on display at each step in a transaction sequence?	0 0 0	
7.5	Are prompts, cues, and messages placed where the eye is likely to be looking on the screen?	0 0 0	
7.6	Have prompts been formatted using white space, justification, and visual cues for easy scanning?	0 0 0	
7.7	Do text areas have "breathing space" around them?	0 0 0	
7.8	Is there an obvious visual distinction made between "choose one" menu and "choose many" menus?	0 0 0	
7.9	Have spatial relationships between soft function keys (on-screen cues) and keyboard function keys been preserved?	0 0 0	
7.10	Does the system gray out or delete labels of currently inactive soft function keys?	0 0 0	
7.11	Is white space used to create symmetry and lead the eye in the appropriate direction?	0 0 0	
7.12	Have items been grouped into logical zones, and have headings been used to distinguish between zones?	0 0 0	
7.13	Are zones no more than twelve to fourteen characters wide and six to seven lines high?	0 0 0	
7.14	Have zones been separated by spaces, lines, color, letters, bold titles, rules lines, or shaded areas?	0 0 0	
7.15	Are field labels close to fields, but separated by at least one space?	0 0 0	
7.16	Are long columnar fields broken up into groups of five, separated by a blank line?	0 0 0	
7.17	Are optional data entry fields clearly marked?	0 0 0	
7.18	Are symbols used to break long input strings into "chunks"?	0 0 0	
7.19	Is reverse video or color highlighting used to get the user's attention?	0 0 0	

7.20	Is reverse video used to indicate that an item has been selected?	0 0 0	
7.21	Are size, boldface, underlining, color, shading, or typography used to show relative quantity or importance of different screen items?	0 0 0	
7.22	Are borders used to identify meaningful groups?	0 0 0	
7.23	Has the same color been used to group related elements?	0 0 0	
7.24	Is color coding consistent throughout the system?	0 0 0	
7.25	Is color used in conjunction with some other redundant cue?	0 0 0	
7.26	Is there good color and brightness contrast between image and background colors?	0 0 0	
7.27	Have light, bright, saturated colors been used to emphasize data and have darker, duller, and desaturated colors been used to de-emphasize data?	0 0 0	
7.28	Is the first word of each menu choice the most important?	0 0 0	
7.29	Does the system provide <i>mapping</i> : that is, are the relationships between controls and actions apparent to the user?	0 0 0	
7.30	Are input data codes distinctive?	0 0 0	
7.31	Have frequently confused data pairs been eliminated whenever possible?	0 0 0	
7.32	Have large strings of numbers or letters been broken into chunks?	0 0 0	
7.33	Are inactive menu items grayed out or omitted?	0 0 0	
7.34	Are there menu selection defaults?	0 0 0	
7.35	If the system has many menu levels or complex menu levels, do users have access to an on-line spatial menu map?	0 0 0	
7.36	Do GUI menus offer affordance: that is, make obvious where selection is possible?	0 0 0	
7.37	Are there salient visual cues to identify the active window?	0 0 0	
7.38	Are function keys arranged in logical groups?	0 0 0	
7.39	Do data entry screens and dialog boxes indicate when fields are	0 0 0	

	optional?		
7.40	On data entry screens and dialog boxes, are dependent fields displayed only when necessary?	0 0 0	

A.5.8. Flexibility and Minimalist Design

Accelerators-unseen by the novice user-may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. Provide alternative means of access and operation for users who differ from the "average" user (e.g., physical or cognitive ability, culture, language, etc.)

#	Review Checklist	Yes No N/A	Comments
8.1	If the system supports both novice and expert users, are multiple levels of error message detail available?	0 0 0	
8.2	Does the system allow novices to use a keyword grammar and experts to use a positional grammar?	0 0 0	
8.3	Can users define their own synonyms for commands?	0 0 0	
8.4	Does the system allow novice users to enter the simplest, most common form of each command, and allow expert users to add parameters?	0 0 0	
8.5	Do expert users have the option of entering multiple commands in a single string?	0 0 0	
8.6	Does the system provide function keys for high-frequency commands?	0 0 0	
8.7	For data entry screens with many fields or in which source documents may be incomplete, can users save a partially filled screen?	0 0 0	
8.8	Does the system automatically enter leading zeros?	0 0 0	
8.9	If menu lists are short (seven items or fewer), can users select an item by moving the cursor?	0 0 0	

8.10	If the system uses a type-ahead strategy, do the menu items have mnemonic codes?	0 0 0	
8.11	If the system uses a pointing device, do users have the option of either clicking on fields or using a keyboard shortcut?	0 0 0	
8.12	Does the system offer "find next" and "find previous" shortcuts for database searches?	0 0 0	
8.13	On data entry screens, do users have the option of either clicking directly on a field or using a keyboard shortcut?	0 0 0	
8.14	On menus, do users have the option of either clicking directly on a menu item or using a keyboard shortcut?	0 0 0	
8.15	In dialog boxes, do users have the option of either clicking directly on a dialog box option or using a keyboard shortcut?	0 0 0	
8.16	Can expert users bypass nested dialog boxes with either type-ahead, user-defined macros, or keyboard shortcuts?	0 0 0	

A.5.9. Aesthetic and Minimalist Design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

#	Review Checklist	Yes No N/A	Comments
9.1	Is only (and all) information essential to decision making displayed on the screen?	0 0 0	
9.2	Are all icons in a set visually and conceptually distinct?	0 0 0	
9.3	Have large objects, bold lines, and simple areas been used to distinguish icons?	0 0 0	
9.4	Does each icon stand out from its background?	0 0 0	
9.5	If the system uses a standard GUI interface where menu sequence has already been specified, do menus adhere to the specification	0 0 0	

	whenever possible?		
9.6	Are meaningful groups of items separated by white space?	O O O	
9.7	Does each data entry screen have a short, simple, clear, distinctive title?	O O O	
9.8	Are field labels brief, familiar, and descriptive?	O O O	
9.9	Are prompts expressed in the affirmative, and do they use the active voice?	O O O	
9.10	Is each lower-level menu choice associated with only one higher level menu?	O O O	
9.11	Are menu titles brief, yet long enough to communicate?	O O O	
9.12	Are there pop-up or pull-down menus within data entry fields that have many, but well-defined, entry options?	O O O	

A.5.10. Help and Documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

#	Review Checklist	Yes No N/A	Comments
10.1	If users are working from hard copy, are the parts of the hard copy that go on-line marked?	O O O	
10.2	Are on-line instructions visually distinct?	O O O	
10.3	Do the instructions follow the sequence of user actions?	O O O	
10.4	If menu choices are ambiguous, does the system provide additional explanatory information when an item is selected?	O O O	
10.5	Are data entry screens and dialog boxes supported by navigation and completion instructions?	O O O	
10.6	If menu items are ambiguous, does the system provide additional	O O O	

	explanatory information when an item is selected?		
10.7	Are there memory aids for commands, either through on-line quick reference or prompting?	O O O	
10.8	Is the help function visible; for example, a key labeled HELP or a special menu?	O O O	
10.9	Is the help system interface (navigation, presentation, and conversation) consistent with the navigation, presentation, and conversation interfaces of the application it supports?	O O O	
10.10	Navigation: Is information easy to find?	O O O	
10.11	Presentation: Is the visual layout well designed?	O O O	
10.12	Conversation: Is the information accurate, complete, and understandable?	O O O	
#	Review Checklist	Yes No N/A	Comments
10.13	Is the information relevant?	O O O	
10.14	Goal-oriented (What can I do with this program?)	O O O	
10.15	Descriptive (What is this thing for?)	O O O	
10.16	Procedural (How do I do this task?)	O O O	
10.17	Interpretive (Why did that happen?)	O O O	
10.18	Navigational (Where am I?)	O O O	
10.19	Is there context-sensitive help?	O O O	
10.20	Can the user change the level of detail available?	O O O	
10.21	Can users easily switch between help and their work?	O O O	
10.22	Is it easy to access and return from the help system?	O O O	
10.23	Can users resume work where they left off after accessing help?	O O O	

A.5.11. Skills

The system should support, extend, supplement, or enhance the user's skills, background knowledge, and expertise ----not replace them.

#	Review Checklist	Yes No N/A	Comments
11.1	Can users choose between iconic and text display of information?	O O O	
11.2	Are window operations easy to learn and use?	O O O	
11.3	If users are experts, usage is frequent, or the system has a slow response time, are there fewer screens (more information per screen)?	O O O	
11.4	If users are novices, usage is infrequent, or the system has a fast response time, are there more screens (less information per screen)?	O O O	
11.5	Does the system automatically color-code items, with little or no user effort?	O O O	
11.6	If the system supports both novice and expert users, are multiple levels of detail available.	O O O	
11.7	Are users the initiators of actions rather than the responders?	O O O	
11.8	Does the system perform data translations for users?	O O O	
11.9	Do field values avoid mixing alpha and numeric characters whenever possible?	O O O	
11.10	If the system has deep (multilevel) menus, do users have the option of typing ahead?	O O O	
11.12	When the user enters a screen or dialog box, is the cursor already positioned in the field users are most likely to need?	O O O	
11.13	Can users move forward and backward within a field?	O O O	
11.14	Is the method for moving the cursor to the next or previous field both simple and visible?	O O O	
11.15	Has auto-tabbing been avoided except when fields have fixed lengths or users are experienced?	O O O	
11.16	Do the selected input device(s) match user capabilities?	O O O	

11.17	Are cursor keys arranged in either an inverted T (best for experts) or a cross configuration (best for novices)?	O O O	
11.18	Are important keys (for example, <u>ENTER</u> , <u>TAB</u>) larger than other keys?	O O O	
11.19	Are there enough function keys to support functionality, but not so many that scanning and finding are difficult?	O O O	
11.20	Are function keys reserved for generic, high-frequency, important functions?	O O O	
11.21	Are function key assignments consistent across screens, subsystems, and related products?	O O O	
11.22	Does the system correctly anticipate and prompt for the user's probable next activity?	O O O	

A.5.12. Pleasurable and Respectful Interaction with the User

The user's interactions with the system should enhance the quality of her or his work-life. The user should be treated with respect. The design should be aesthetically pleasing- with artistic as well as functional value.

#	Review Checklist	Yes No N/A	Comments
12.1	Is each individual icon a harmonious member of a family of icons?	O O O	
12.2	Has excessive detail in icon design been avoided?	O O O	
12.3	Has color been used with discretion?	O O O	
12.4	Has the amount of required window housekeeping been kept to a minimum?	O O O	
12.5	If users are working from hard copy, does the screen layout match the paper form?	O O O	
12.6	Has color been used specifically to draw attention, communicate organization, indicate status changes, and establish relationships?	O O O	
12.7	Can users turn off automatic color coding if necessary?	O O O	

12.8	Are typing requirements minimal for question and answer interfaces?	O O O	
12.9	Do the selected input device(s) match environmental constraints?	O O O	
12.13	If the system uses multiple input devices, has hand and eye movement between input devices been minimized?	O O O	
12.14	If the system supports graphical tasks, has an alternative pointing device been provided?	O O O	
12.15	Is the numeric keypad located to the right of the alpha key area?	O O O	
12.16	Are the most frequently used function keys in the most accessible positions?	O O O	
12.17	Does the system complete unambiguous partial input on a data entry field?	O O O	

A.5.13 Privacy

The system should help the user to protect personal or private information- belonging to the user or the his/her clients.

#	Review Checklist	Yes No N/A	Comments
13.1	Are protected areas completely inaccessible?	O O O	
13.2	Can protected or confidential areas be accessed with certain passwords.	O O O	
13.3	Is this feature effective and successful.	O O O	

System Title: _____ Release #: _____

Evaluator: _____ Date: _____

Figure 17: Detailed Checklist for Usability Inspection

(Retrieved from: <http://www.stcsig.org/usability/topics/articles/he-checklist.html>)

A.5 REFERENCES

- Law, E., Hvannberg, E. and Cockton, G. (2008). Model-Based Evaluation: A New Way to Support Usability Evaluation of Multimodal Interactive Applications. In Bernhaupt, R., D. Navarre, P. Palanque, and M. Winckler. *Maturing Usability*. Springer, pp. 96-122.
- Brinck, T., Gergle, D. and Wood, S. (2002) *Usability for the Web: Designing Web Sites that Work*. Morgan Kaufmann Publishers, CA.
- Law, E., Hvannberg, E. and Cockton, G. (2008). Characterizations, Requirements, and Activities of User-Centered Design. In Jokela, T. *Maturing Usability*. Springer, pp. 168-196.
- Nielsen, J., and Landauer, T. K. (1993). A mathematical model of the finding of usability problems. *Proceedings ACM/IFIP INTERCHI'93 Conference*. Amsterdam, The Netherlands, April 24-29, pp. 206-213.
- Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. *Proc. ACM CHI'94 Conf*. Boston, MA, April 24-28, pp. 152-158.
- Nielsen, J. (1992). Finding usability problems through heuristic evaluation. *Proceedings ACM CHI'92 Conference*. Monterey, CA, May 3-7, pp. 373-380.
- Nielsen, J. (2001). Heuristic Evaluation. Retrieved Dec. 3, 2011, from <http://www.useit.com/papers/heuristic/>.
- Nielsen, J., Loranger, H. (2006). *Prioritizing Web Usability*. New Riders, Berkeley, CA.
- Nielsen, J. (1994). *Usability Inspection Methods*, John Wiley & Sons, New York, NY.
- Pierotti, D. (1993). Usability Techniques. Retrieved Dec. 1, 2011, from <http://www.stcsig.org/usability/topics/articles/he-checklist.html>
- Usability First. (2011). Retrieved Dec. 3, 2011, from <http://www.usabilityfirst.com/glossary/>.

APPENDIX B: REFERENCES

- Cloutier, R., Mostashari, A., McComb, S., Deshmukh, A., Wade, J., Kennedy, D., & Korfiatis, P. (2009). Investigation of a Graphical CONOPS Development Environment for Agile Systems Engineering: Systems Engineering Research Center.
- Cloutier, R., Mostashari, A., McComb, S., Deshmukh, A., Wade, J., Kennedy, D., . . . Carrigy, A. (2010). Investigation of a Graphical CONOPS Development Environment for Agile Systems Engineering: Systems Engineering Research Center.
- Mostashari, A., McComb, S. A., Kennedy, D. M., Cloutier, R., & Korfiatis, P. (2011). Developing a Stakeholder-Assisted Agile CONOPS Development Process. *Systems Engineering*, 15(1), 1-13. doi: 10.1002/sys.20190